UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Viljem Dernikovič

Pozicioniranje v zaprtih prostorih z uporabo NFC tehnologije

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentorica: doc. dr. Ganna Kudryavtseva

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomskega dela preučite pozicioniranje v zaprtih prostorih z uporabo NFC tehnologije. Za ta namen sestavite večnivojski sistem, ki vključuje mobilnega odjemalca. Predstavite probleme, ki se pri tem pojavijo, in poskušajte najti način za njihovo odpravo. Izberite problem in izvedite meritve ter s statistično analizo ovrednotite rezultate.

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Viljem Dernikovič,

z vpisno številko 63020029,

sem avtor diplomskega dela z naslovom:

Pozicioniranje v zaprtih prostorih z uporabo NFC tehnologije

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Ganne Kudryavtseve,

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 6. junija 2014 Podpis avtorja:

Zahvala

Zahvaljujem se družini, ki me je v času študija spodbujala in mi stala ob strani ter s potrpežljivostjo prenašala mojo odsotnost. Zahvaljujem se tudi mentorici doc. dr. Ganni Kudryavtsevi za pomoč in nasvete pri tem diplomskem delu. Posebna zahvala tudi Boštjanu Gomilšku iz podjetja Digicom d.o.o. za spodbude in praktične primere.

Kazalo

Povzetek 1

Abstract 2

1 **Uvod** 3

2 Določanje trenutnega položaja 5
2.1 Sateliti 5
2.2 Mobilna omrežja 5
2.3 Brezžična lokalna omrežja (angl. Wifi) 5
2.4 Ultra-širokopasovna tehnologija (angl. Ultra-wideband) 6

3 Sistem za pozicioniranje v zaprtih prostorih 7
3.1 NFC tehnologija 7
3.2 Opis delovanja sistema 11
3.3 Matematično in fizikalno ozadje 15
3.4 Dijkstrov algoritem 19

4 Razvoj sistema 21
4.1 Strojna oprema 21
4.2 Pospeškomer 21
4.3 Magnetometer 22
4.4 Žiroskop 22
4.5 Programska oprema 22
4.6 Arhitektura sistema 23
4.7 Podatkovne strukture 26
4.8 Mobilna aplikacija 29
4.9 Strežniška aplikacija 30
4.10 Podatkovna baza 31
4.11 Primer aplikacije za vnos zemljevida 33

5 Analiza vpliva magnetnega polja 34
5.1 Meritve 35
5.2 Postopki in izračuni 36

6 Sklepne ugotovitve 41

Priloge 43
Priloga A – Mobilna aplikacija 43
Priloga B – Strežniška aplikacija 45
Priloga C – Aplikacija za vnos zemljevida 47
Seznam slik 49
Seznam tabel 49
Seznam grafov 50
Seznam izsekov izvorne kode 50

Viri 51

Seznam uporabljenih kratic

API	(angl. Application Programming Interface) programski vmesnik
AR	(angl. Augmented Reality) obogatena resničnost
GPS	(angl. Global Positioning System) sistem globalnega določanja položaja
HUD	(angl. Heads-up Display) prozoren zaslon, ki vidni sliki dodaja podatke
JSON	(angl. JavaScript Object Notation) format podatkov osnovan na JavaScript-u
NFC	(angl. Near Field Communication) sistem za bližnjo komunikacijo
POI	(angl. Point Of Interest) zanimiva točka
REST	(angl. Representational State Transfer) arhitektura za izmenjavo podatkov med spletnimi storitvami
RSS	(angl. Received Signal Strength) jakost sprejetega signala
RSSI	(angl. Received Signal Strength Indicator) indikator jakosti sprejetega signala
SOA	(angl. Service Oriented Architecture) storitveno usmerjena arhitektura
UML	(angl. Unified Modeling Language) poenoteni jezik modeliranja
WCF	(angl. Windows Communication Foundation) skupek orodij za implementacijo storitveno usmerjene arhitekture
WiFi	(angl. Wireless Fidelity) brezžično lokalno računalniško omrežje
XML	(angl. Extensible Markup Language) razširljiv označevalni jezik

Povzetek

V tem diplomskem delu je predstavljen primer sistema za pozicioniranje v zaprtih prostorih z uporabo NFC tehnologije. Znano je, da v zaprtih prostorih signal satelitov navigacijskim napravam ni viden, zato pozicioniranje, kot je npr. preko GPSa, ni mogoče. Delo obsega razvoj več aplikacij, vključno z Android aplikacijo na mobilnem telefonu. Uporabnik s pomočjo aplikacije na mobilnem telefonu prebere enolični identifikator iz NFC nalepke in iz strežnika pridobi trenutni položaj in seznam možnih ciljev. Nato lahko izbere cilj in aplikacija ga usmerja po najkrajši poti do cilja preko vmesnih NFC točk, ki jih na tej poti uporabnik lahko prebere. Na zaslonu telefona se prikazujejo informacije, kot so smer, oddaljenost do naslednje točke, celotna oddaljenost do cilja in trenutna hitrost. V delu so opisani matematični in fizikalni problemi in njihove rešitve. V zadnjem poglavju je bila izdelana statistična analiza, ki pripomore pri ugotavljanju anomalij vpliva močnega magnetnega polja.

Ključne besede

pozicioniranje, NFC, hipoteza, senzorsko zlivanje, magnetometer, pospeškomer, žiroskop, Android, grafna podatkovna baza, Neo4J, Kalmanov filter, JSON, Dijkstra

Abstract

In this diploma thesis an example of the indoor positioning system using NFC technology is presented. The indoor signal of satellites, such as GPS, is not visible to the navigational devices, so the positioning there is not possible. The thesis covers the development of several applications including Android application for smartphones. The smartphone user reads ID from NFC label and gets his current position and the list of all possible destinations from the server. User can then select one destination and can receive the shortest path from the current position to the destination from the server. Mobile application navigates him between NFC labels on this shortest path until the destination is reached. The thesis also covers some mathematical and physical problems connected with the topic and possible solutions. In the last chapter the statistical analysis that can help find strong magnetic field anomalies is presented.

Key words

positioning, NFC, hypothesis, sensor fusion, magnetometer, accelerometer, gyroscope, Android, graph database, Neo4J, Kalman filter, JSON, Dijkstra

1 Uvod

V zadnjem času postajajo pametni mobilni telefoni vse bolj razširjeni med prebivalstvom. Ne samo, da mu nudijo osnovne komunikacijske možnosti, kot so telefoniranje in pošiljanje sporočil, temveč tudi brskanje po spletu, pošiljanje elektronski sporočil, komuniciranje preko socialnih omrežjih ipd. Pametni mobilni telefoni ne predstavljajo samo funkcije razširjenega sporočanja, so veliko več. Posamezniku lahko nudijo pomembne informacije v realnem času iz različnih področji, npr. informacijo o trenutni lokaciji, stanje na bančnem računu, preko zunanjih senzorjev zdravstvene podatke (srčni utrip, pritisk, nivo sladkorja v krvi ipd.), spremljanje telemetričnih podatkov in video nadzor v pametni hiši, spremljajo dogajanje v avtomobilu in opozarjajo uporabnika na nevarnosti, v trgovini najdejo najboljši izdelek po najnižji ceni ipd. Torej na eni strani olajšujejo življenje ljudi, na drugi strani pa lahko negativno vplivajo na posameznika, če sam telefon ali nameščene aplikacije niso v skladu z raznimi standardi. Ti standardi postajajo s časom strožji, proizvajalcem telefonov narekujejo večjo zanesljivost, natančnost, manjši vpliv na okolje in zdravje ljudi (npr. SAR vrednost in onesnaževanje ob proizvodnji in razgradnji), razvijalcem aplikacij pa narekujejo razvoj bolj intuitivnih in uporabnikom prijaznih grafičnih vmesnikov, skrb za varnost podatkov in zasebnost ter učinkovito delovanje.

V tem diplomskem delu se bom posvetil razvoju celovite rešitve, ki uporabniku s pametnim mobilnim telefonom omogoča, da najde svojo trenutno lokacijo v zaprtem prostoru, kjer klasično pozicioniranje ne deluje, in ga usmerja po najkrajši poti do končne izbrane točke. Klasično pozicioniranje je pozicioniranje predvsem preko GPS signala. Zanj je značilno, da ne deluje povsod, predvsem v zaprtih prostorih, gosto poseljenih območjih, pod zemljo in še kje. Na drugi strani se odpira tudi vrsta ostalih tehnologij, ki rešujejo ta problem, vendar so v praksi bodisi težko izvedljive ali preveč drage bodisi premalo natančne ali pa kombinacija slednjih. Osredotočil se bom predvsem na tehnologijo, ki uporablja NFC nalepke in interne senzorje telefona, kot so pospeškomer, magnetometer in žiroskop, ki skupaj z aplikacijo lahko tvorijo dokaj natančen inercijski navigacijski sistem. V današnje pametne telefone je vsa strojna oprema praktično že vgrajena, ne samo v telefone višjega, temveč tudi v telefone nižjega cenovnega razreda, ki jih uporablja večina ljudi. Trend razvoja mobilnih telefonov kaže, da bo ta tudi v prihodnosti del nabora komponent v telefonih, kar pomeni, da bo cena telefonov še nižja in tako v končni fazi dosegljiva vsem. Verjetno se bo v bližnji prihodnosti tudi formiral standard, ki bo urejal to področje kot celoto. Sedaj so namreč komponente že del posameznih standardov, npr. NFC standard ISO/IEC 14443 in ostali, ki omogočajo kompatibilnost različnih vrst naprav različnih proizvajalcev. Sam menim, da bo področje vsaj toliko pomembno, kot je

sedaj navigacija, kjer je GPS signal dostopen, če ne še bolj, saj se ljudje večino časa nahajajo znotraj stavb in poleg osnovnega poslanstva, torej pozicioniranja v prostoru, odpira še vrsto drugih možnosti kombiniranja z ostalimi aplikacijami in napravami, npr. iskanje izdelkov in marketing glede na trenutno lokacijo v trgovini, iskanje oseb, predmetov, oddelkov, organizacijskih enot, na splošno zanimivih točk (angl. POI) znotraj stavb. Vsekakor pokriva širok spekter področij, kjer se rešitev lahko uporablja, npr. zdravstvo, logistika, trgovina, policija, reševanje, energetika, lesna industrija, transport, turizem ipd., kot tudi različne naprave, ki niso ravno pametni telefoni, npr. očala za prikaz obogatene resničnosti (angl. AR) ali naprave tipa HUD (angl. Heads-up Display).

2 Določanje trenutnega položaja

Problem določanja trenutnega položaja predmetov in oseb izvira že iz zgodovinskih časov, ko so ljudje pričeli z raziskovanjem Zemlje in se je nadaljevalo pri odkrivanju vesolja. Pomen določanja pravilnega trenutnega položaja je bil vedno pomemben, saj je velikokrat pomenil ločnico med življenjem in smrtjo popotnikov, raziskovalcev, vojakov in ostalih, da so lahko pravočasno prišli s točke A na točko B ali našli osebo oz. predmet v nekem prostoru. Danes poznamo vrsto tehnologij, ki nam z določeno natančnostjo postrežejo ta podatek, in se uporabljajo za različne namene.

2.1 Sateliti

Trenutno se omenjajo 4 največji satelitski sistemi [1]:

- ameriški US Navstar Global Positioning System (angl. GPS)
- ruski GLONASS
- evropski GALILEO
- kitajski COMPASS* (v nastajanju)

Bistvena prednost je visoka natančnost pošiljanja podatkov o zemljepisni širini in dolžini ter višini, slabost pa je, da signal iz satelitov do sprejemnika motijo zidovi in strehe, kar pomeni, da sistem ne deluje v zaprtih prostorih.

2.2 Mobilna omrežja

Signal mobilnega omrežja brez težav pride do telefona tudi znotraj stavb. Pozicioniranje telefona temelji na signalu, ki ga v določenem času pošlje bazna postaja. Telefon lahko identificira bazno postajo, ki jo v tistem trenutku uporablja. Dejstvo, da je lahko bazna postaja tudi 35 km daleč, pomeni, da natančnost določanja položaja ni ravno visoka.

2.3 Brezžična lokalna omrežja (angl. Wifi)

WiFi (angl. Wireless Fidelity) je skupno ime za standard IEEE 802.11. Brezžična omrežja so danes praktično prisotna povsod. Brezžični usmerjevalnik oddaja signal, ki ga sprejemajo naprave v njegovem dosegu. Te naprave imajo sposobnost zaznavanja jakosti tega signala. Indikator jakosti sprejetega signala (angl. RSSI) je dimenzijska metrika, ki služi za primerjavo med različnimi brezžičnimi usmerjevalniki. Tukaj je potrebno opozoriti, da ne obstaja standard, ki bi direktno pretvarjal RSSI v dejansko jakost sprejetega signala (angl. RSS), torej v enoto

dBm, ampak imajo proizvajalci svoje sheme za pretvorbo. Iz jakosti sprejetega signala in njegove frekvence se lahko izračuna razdalja po različnih matematičnih modelih, ki jih tukaj zaradi obsežnosti področja ne opisujemo.

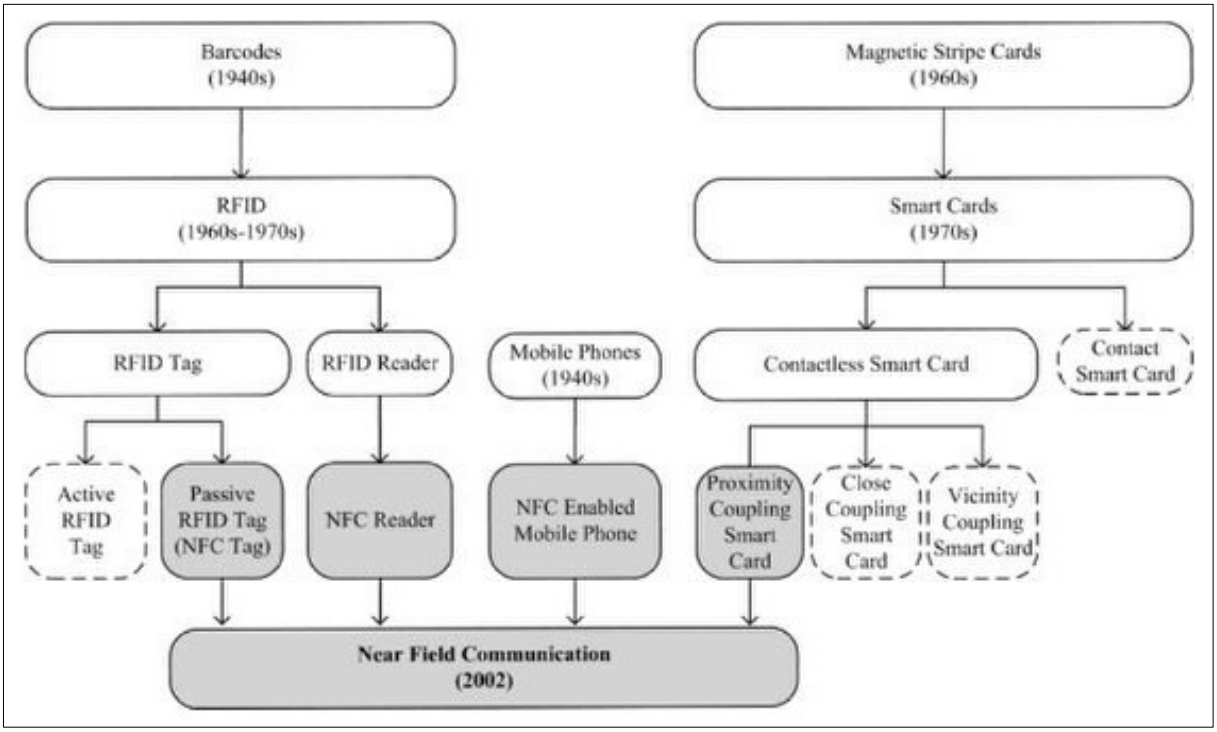
2.4 Ultra-širokopasovna tehnologija (angl. Ultra-wideband)

Signali te vrste ponujajo do centimetra natančno informacijo o lokaciji [2]. Prednosti so tudi v majhni porabi energije in širokopasovnosti, ki prinaša večjo zanesljivost. Uporaba širokega nabora frekvenc povečuje verjetnost, da bo signal zaobšel oviro. Poleg tega je celoten sistem manj dovzeten za motnje oz. interferenco, ki jo povzročajo ostali radijski signali. Problem pri tem načinu so visoki stroški vzpostavitve dovolj velike mreže postaj, ki oddajajo tak signal.

3 Sistem za pozicioniranje v zaprtih prostorih

3.1 NFC tehnologija

NFC (angl. Near Field Communication) je visokofrekvenčna komunikacijska tehnologija kratkega dosega, ki omogoča izmenjavo podatkov na razdalji do 10 cm. [3]



Slika 1: Zgodovina razvoja NFC tehnologije [4]

Zgornja slika (Slika 1) prikazuje zgodovino razvoja NFC tehnologije. Kot je iz slike razvidno, sama tehnologija izhaja iz RFID (angl. Radio-frequency identification) tehnologije, ki se je razvila že v 70. letih prejšnjega stoletja. RFID deluje v različnih frekvenčnih območjih. Podrobno jih opisuje množica standardov in protokolov. V tabeli (Tabela 1) so prikazana frekvenčna območja in njihove razdalje delovanja.

RFID frekvenčno območje	Razdalja delovanja
120–150 kHz (LF)	Do 10 cm
13,56 MHz (HF)	Do 1 m
433 MHz (UHF)	Od 1 m do 100 m
865–868 MHz in 902–928MHz (UHF)	1–2 m
2450–5800 MHz (mikrovalovi)	1–2 m
3,1-10 GHz (mikrovalovi)	Do 200 m

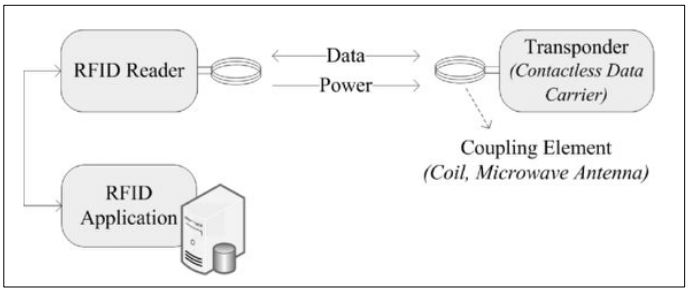
Tabela 1: Frekvenčna območja in razdalje delovanja

NFC deluje v območju 13,56 MHz in razširja visokofrekvenčne (HF) RFID standarde. Razliko med HF RFID in NFC prikazuje spodnja tabela.

	HF RFID	NFC
Frekvenca delovanja	13,56 MHz	13,56 MHz
Komunikacija	enosmerna	obojestranska
Standardi	ISO 14443, 15693, 18000	ISO 14443
Razdalja delovanja	Do 1 m	Do 10 cm
Hkratnost dostopanja	Več dostopov iz ene naprave naenkrat	Samo en dostop iz ene naprave naenkrat

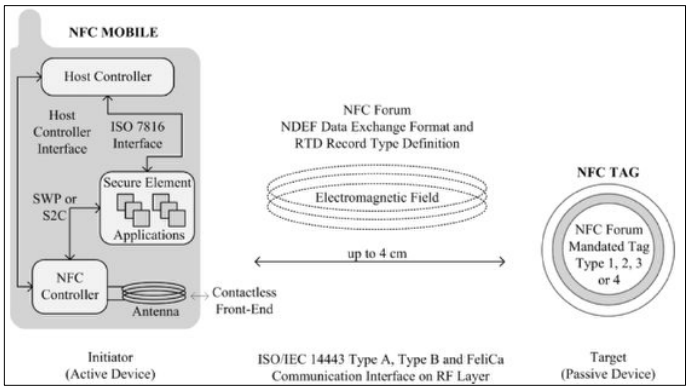
Tabela 2: Primerjava HF RFID in NFC

Delovanje med iniciatorjem in tarčo ponazarja spodnja slika (Slika 2). Iniciator je naprava z lastnim napajanjem (npr. mobilni telefon ali NFC bralnik), t. i. aktivna naprava, tarča pa NFC nalepka (pasivna naprava) ali druga NFC naprava (npr. lahko tudi mobilni telefon), tudi t. i. aktivna naprava, ker ima svoje napajanje. Iniciator vzpostavi dovolj močno elektromagnetno polje in vzpodbudi v tarči dovolj inducirane napetosti, da lahko brezžično pošlje svoje podatke iniciatorju. Na omenjeni sliki je prikazana dvosmerna podatkovna povezava, kar pomeni, da lahko podatki potujejo tudi iz smeri iniciatorja že na samem začetku komunikacije.

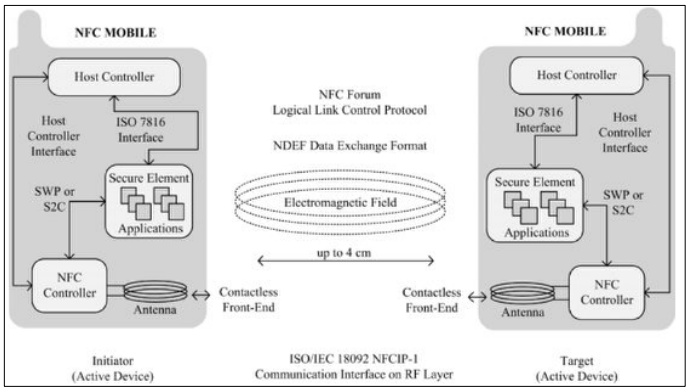


Slika 2: Primer iniciatorja in tarče

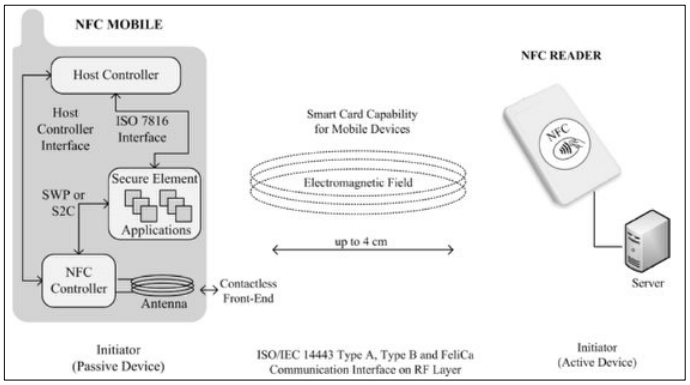
Naslednje slike prikazujejo tri bistvene primere uporabe NFC tehnologije. V našem primeru bomo uporabljali prvi način, torej komunikacijo med mobilnim telefonom in NFC nalepko.



Slika 3: Komunikacija med mobilnim telefonom in nalepko

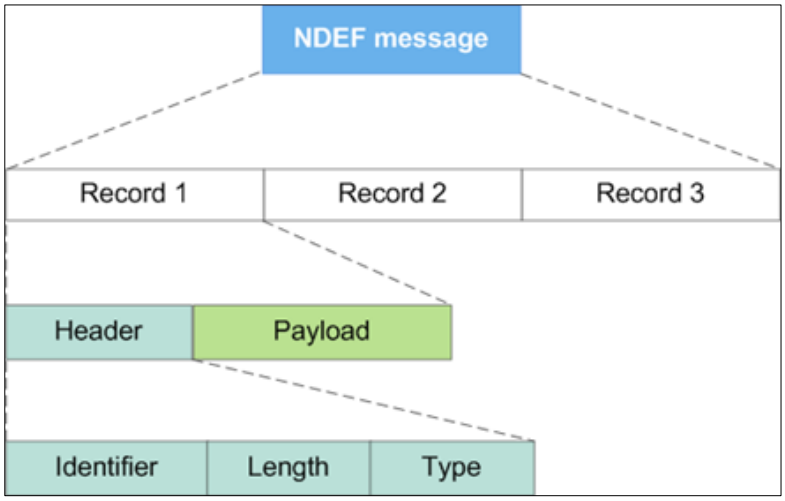


Slika 4: Komunikacija med dvema mobilnima telefonoma

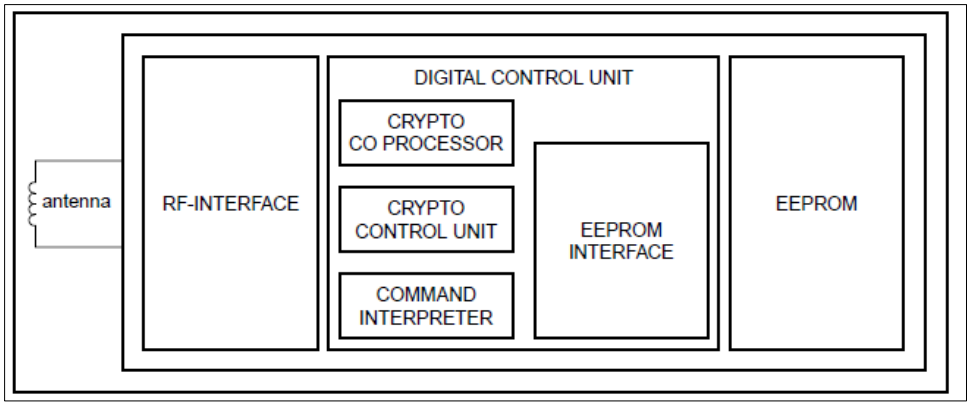


Slika 5: Komunikacija med mobilnim telefonom in bralnikom

NFC nalepka pozna način pisanja in branja podatkov. Sama zgradba podatkov se nahaja v t. i. obliki NDEF (angl. NFC Data Exchange Format).



Slika 6: Format za izmenjavo podatkov, ki jo uporablja NFC tehnologija



Slika 7: Blok diagram Mifare Ultralight C nalepke

Page address		Byte number			
Decimal	Hex	0	1	2	3
0	00h	serial number			
1	01h	serial number			
2	02h	serial number	internal	lock bytes	lock bytes
3	03h	OTP	OTP	OTP	OTP
4 to 39	04h to 27h	user memory	user memory	user memory	user memory
40	28h	lock bytes	lock bytes	-	-
41	29h	16-bit counter	16-bit counter	-	-
42	2Ah	authentication configuration			
43	2Bh	authentication configuration			
44 to 47	2Ch to 2Fh	authentication key			

Slika 8: Organizacija pomnilnika Mifare Ultralight C nalepke

3.2 Opis delovanja sistema

Sedaj, ko smo spoznali delovanje NFC tehnologije, lahko opišemo, kako deluje sistem pozicioniranja v zaprtem prostoru z uporabo te tehnologije. Pred sabo imejmo zemljevid oz. tloris stavbe, za katerega poznamo merilo in smer postavitve zemljevida glede na stran neba. Zgornji rob tlorisa je normalno obrnjen proti severu, desni proti vzhodu, spodnji proti jugu in levi proti zahodu. Na zemljevid postavimo točke, ki bodo služile za postavitev NFC nalepk, zato naj bodo izbrane na strateško pomembnih mestih (npr. na vhodu, ob razpotjih, pred stopniščem, pred prostori, do katerih želimo priti itd.). Med izbranimi točkami narišemo povezave tako, da predstavljajo neovirano pot (npr. pot po hodniku, pot skozi prostore, pot po stopnišču). Pri vsaki povezavi je pomembna tudi smer in pomeni dovoljeno smer hoje. Normalno sta na vsaki povezavi dovoljeni obe smeri (naprej in nazaj), v tem primeru ne rišemo puščic, v določenih primerih, npr. pri tekočih stopnicah, pa je dovoljena le ena smer (npr. samo naprej ali samo nazaj), kar prikažemo s puščico. Točke so postavljene v 2-dimenzionalen koordinatni sistem, z osjo x in y, kot prikazuje spodnja slika (Slika 9). Posamezna točka je določena z oddaljenostjo od osi x in y. Razdalja r med dvema točkama $T_1(x_1, y_1)$ in $T_2(x_2, y_2)$ je enaka:

$$r = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

(1)

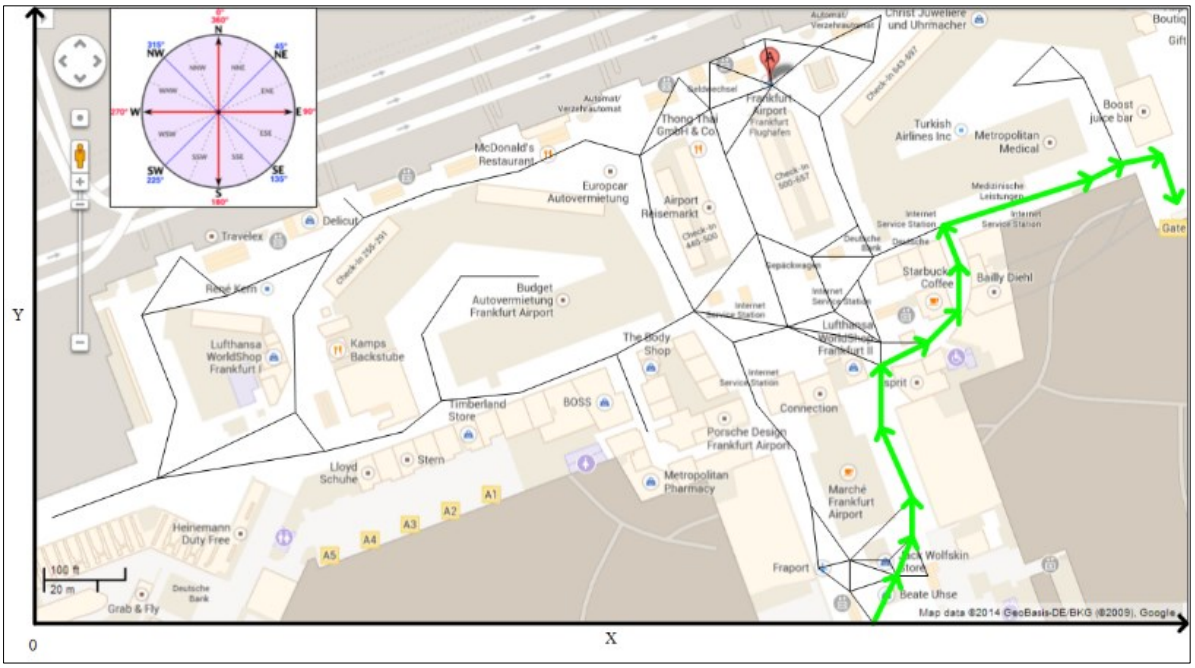
Ker poznamo tudi merilo zemljevida, torej razdalja na zemljevidu (r_p) ustreza določeni razdalji (r_m) v metričnem sistemu (npr. metrih), lahko izračunamo posamezne dolžine vseh povezav, npr. v metrih (r_{xy}). Faktor označimo s F.

$$r_{xy} = r * \frac{r_m}{r_p} = r * F$$

(2)

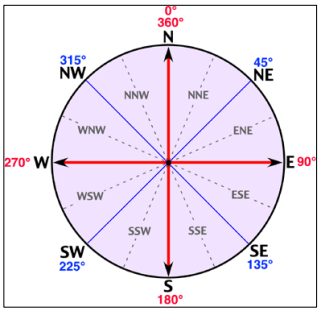
Smeri povezav smo že določili na zemljevidu in izračunali njihove dolžine, sedaj pa lahko določimo še kote med posameznimi povezavami oz. vektorji in navpično osjo oz. enotskim vektorjem $\vec{j} = (0, 1)$. Posamezne kote lahko izračunamo iz skalarnega produkta dveh vektorjev:

$$\alpha = \operatorname{acos}\left(\frac{\vec{r} * \vec{j}}{|\vec{r}| * |\vec{j}|}\right) \tag{3}$$



Slika 9: Primer zemljevida z vrisanimi potmi

Kot bomo kasneje videli, bomo v našem programu uporabljali stopinje v območju od 0° do 360°.

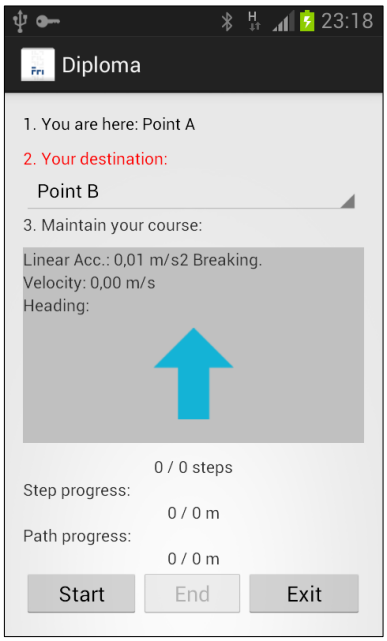


Slika 10: Določanje kota oz. smeri na povezavi

Takšen opisan zemljevid, torej s točkami, povezavami, razdaljami in smermi, se lahko sedaj shrani v posebno grafno podatkovno bazo. V okviru tega diplomskega dela je bil razvit tudi poseben vmesnik, ki zna komunicirati s takšno bazo, torej iskati posamezne točke, najti vse

točke, najti vse končne točke, poiskati najkrajšo oz. najkrajše poti ipd. Ta vmesnik je del strežniškega programa in je preko spletnih storitev dostopen na odjemalcih, torej pametnih mobilnih telefonih, ki imajo nameščen program. Oglejmo si sedaj primer programa na mobilnem telefonu z operacijskim sistemom Android.

Uporabnik vstopi v stavbo in na vidnem mestu opazi NFC nalepko. Ker njegov telefon podpira NFC tehnologijo, lahko prebere vsebino NFC nalepke. Ker smo v prejšnjem razdelku spoznali, da nalepka lahko vsebuje tudi zapis, ki označuje spletno povezavo, lahko telefon uporabnika usmeri direktno na spletno mesto. Tukaj predpostavimo, da ima uporabnik s svojim telefonom dostop do interneta, bodisi preko mobilnega omrežja ali pa lokalnega brezžičnega omrežja. Ker uporabnik še nima nameščenega programa, ga povezava usmeri na spletno stran Google Play, kjer so objavljeni vsi programi za operacijski sistem Android. Od tam si ga lahko uporabnik namesti. Ob ponovnem branju NFC nalepke, se sedaj namesto spletne povezave odpre kar sam program, ki že takoj na začetku uporabniku sporoči, na kateri točki se trenutno nahaja (Slika 11).

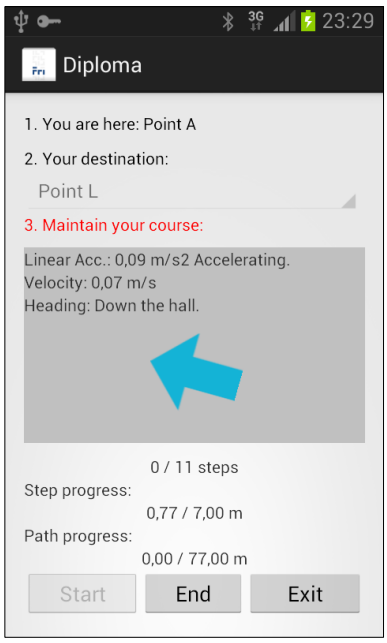


Slika 11: Telefon sporoči trenutno lokacijo uporabnika

Uporabnik nato izbere cilj, kamor bi želel prispeti. Če se npr. odloči obiskati točko L, jo izbere v seznamu in pritisne gumb Start. Iz strežnika se na telefon prenese izračunana najkrajša pot iz točke A v točko L. Pot (angl. path) vsebuje korake (angl. step), ki so v bistvu posamezne povezave med dvema točkama. Na zaslonu so prikazane tudi ostale informacije (Slika 12):

- trenutna hitrost (angl. velocity) v enotah m/s
- opisna smer (angl. heading), npr. pojdi po hodniku
- smer naslednje točke, do katere moramo priti (modra puščica)
- št. doseženih korakov/št. vseh korakov
- razdalja med prejšnjo točko in trenutno lokacijo/razdalja med dvema točkama v metrih

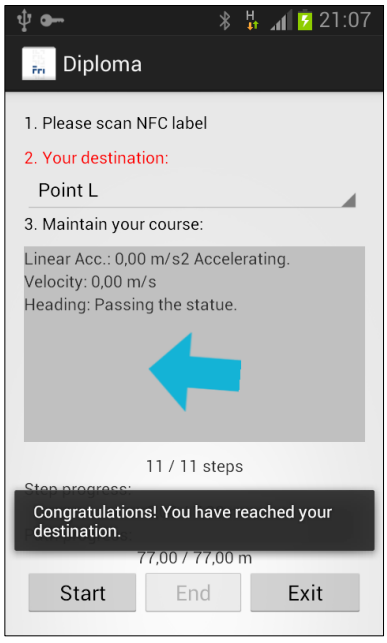
- vsota doseženih korakov/celotna razdalja med začetno in končno točko v metrih



Slika 12: Navigacija

Uporabnik lahko z gumbom End konča navigacijo in na novo poišče najkrajšo pot. Lahko preskakuje korake (naprej in nazaj) v primeru, da ne želi prebrati vmesnih NFC nalepk, ali pa če ga je svobodna volja odnesla izven zastavljene poti in se na poti izgubi. Poseben primer je tudi prihod do stopnišča. Aplikacija mu v tem primeru javi, ali naj izbere pot navzgor ali navzdol.

Ko uporabnik prebere zadnjo NFC nalepko, mu aplikacija javi, da se nahaja na cilju in s tem se navigacija tudi konča (Slika 13).



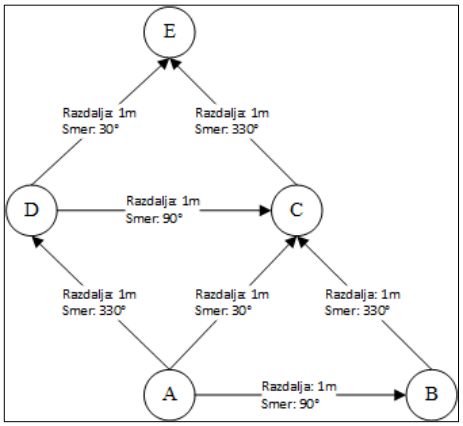
Slika 13: Sporočilo ob prispetju na cilj

3.3 Matematično in fizikalno ozadje

V prejšnjem razdelku smo na zemljevid narisali posebno strukturo, ki jo lahko imenujemo matematični graf. Graf označimo s črko G, množico točk grafa z V(G) in seznam povezav z E(G). Ker je v našem primeru pomembna tudi smer na povezavah, ki jih označujemo s puščicami, lahko rečemo, da gre za usmerjeni graf ali digraf.
Za graf spodaj (Slika 14) velja:

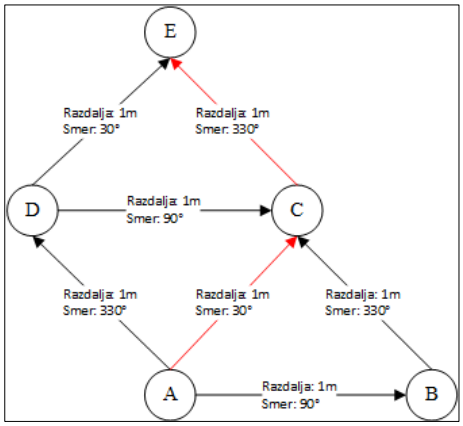
$$V(G) = \{A, B, C, D, E\} \tag{4}$$

$$E(G) = \{AB, AC, AD, BC, CE, DC, DE\} \tag{5}$$



Slika 14: Graf z usmerjenimi povezavami

Na tem mestu lahko definiramo še pojem sprehoda in poti. Sprehod dolžine k v grafu G je zaporedje k povezav grafa G, npr. oblike AB, BC, CE. Sprehod med točkama A in E označimo z ABCE. Če so vse povezave sprehoda različne, potem sprehod imenujemo enostavni sprehod ali sled. Če so v enostavnem sprehodu vse točke različne, potem sprehod imenujemo pot. Kot smo videli v aplikaciji, smo na grafu izvedli algoritem za izračun najkrajše poti. Iz zgornje slike (Slika 14) lahko hitro opazimo, da je v grafu G iz točke A do E lahko več najkrajših poti (npr. ACE in ADE). Strežniški program najde vse najkrajše poti. Uporabnik na mobilnem telefonu bi zato moral izbrati eno od poti, vendar pa zaradi poenostavljanja privzamemo, da se upošteva samo prva pot, ki jo algoritem najde.



Slika 15: Z rdečo barvo označena najkrajša pot grafa G

Iz zgornje slike (Slika 15) vidimo, da iz točke A do E moramo prehoditi najmanj 2 metra. Pri tem nas program v vsakem trenutku obvešča, kakšna je naša trenutna hitrost, kolikšno razdaljo v metrih smo že prehodili, koliko znaša celotna pot ipd. Poglejmo sedaj, kako z internimi senzorji telefona, kot so pospeškomer, senzor magnetnega polja in žiroskop, izračunamo vse prikazane parametre.

V našem vsakdanjem življenju se srečujemo z gibanjem v prostoru, kjer je čas absolutna spremenljivka. Torej pogosto nas zanima, če se v času $t_0 = 0$ nahajamo v točki $T_0(x, y, z)$ in se znajdemo v času $t_1 = t$ v točki $T_1(x, y, z)$, kolikšno pot smo prehodili, kakšna je bila naša povprečna hitrost in kako se je v tem času spreminjala naša hitrost, kar opisujemo s pospeškom. V tem diplomskem delu nas zanimajo fizikalne količine le v ravnini, čeprav iz predhodno naštetih senzorjev pridobivamo tudi koordinato z, ki pa jo v izračunih poskušamo izničiti. Torej zanima nas razdalja (označimo s črko r) od točke $T_0(x, y)$ do točke $T_1(x, y)$, povprečna hitrost (označimo s črko v) na tej poti in pospešek (označimo s črko a). Če smo še bolj natančni, odčitek iz senzorja pospeška vključuje tudi gravitacijski pospešek, zato ga želimo izločiti, ker nas pravzaprav zanima samo linearni pospešek, torej kakšna je sprememba hitrosti v nekem času in v določeni smeri v ravnini (ta ravnina je npr. tloris stavbe).

$$a = \frac{dv}{dt} = \frac{d}{dt}\left(\frac{dr}{dt}\right) = \frac{d^2r}{dt^2} \tag{6}$$

$$v = \frac{dr}{dt} \tag{7}$$

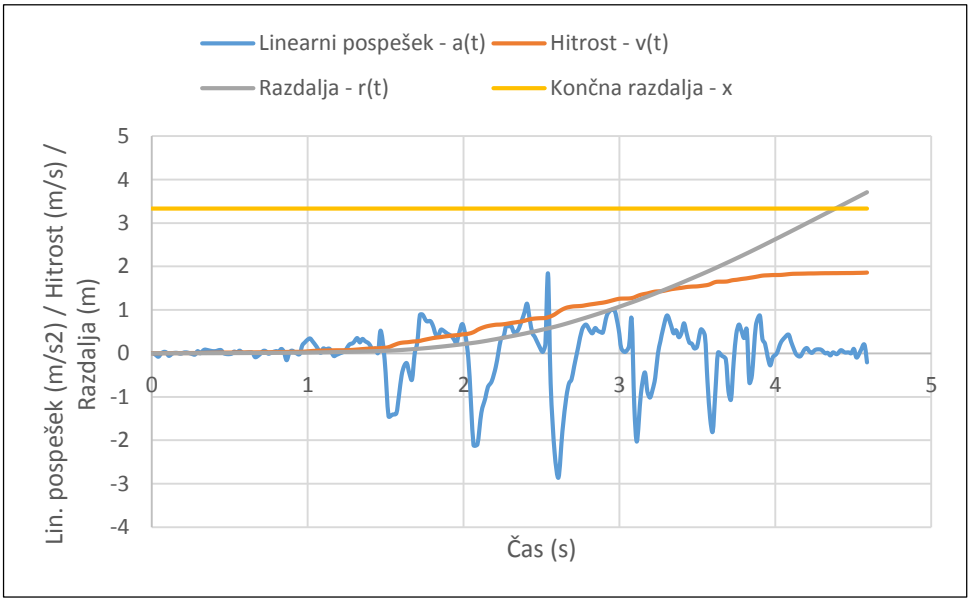
Iz linearnega pospeška, ki ga pridobimo iz senzorja in meritve časa, lahko z integriranjem izračunamo hitrost:

$$v(t) = v_0 + a\int_0^t dt = v_0 + at \tag{8}$$

S ponovnim integriranjem pridemo do razdalje:

$$r(t) = \int_0^t (v_0 + at)dt = v_0\int_0^t dt + a\int_0^t tdt = v_0\,t + \frac{at^2}{2} \tag{9}$$

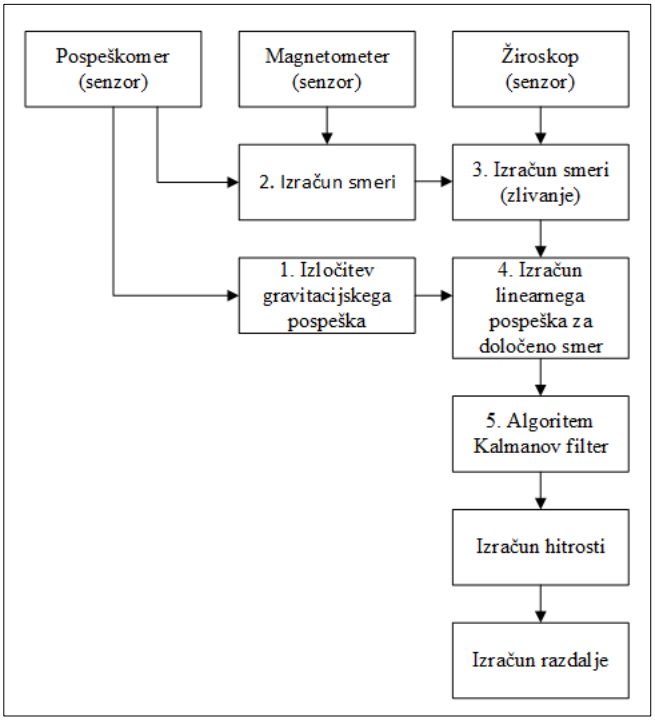
V idealnem sistemu, kjer ni šuma v senzorskih podatkih, bi zgornji izračun razdalje že zadostoval oz. bi prišli do točne razdalje. V realnosti, kjer je prisoten šum, pa z zgornjim načinom izračuna razdalje pridelamo v določenem času tako veliko napako, da je podatek o razdalji popolnoma neuporaben.



Graf 1: Linearni pospešek, hitrost in razdalja v odvisnosti od časa

Iz grafa (Graf 1) vidimo, da se pri 4,5 sekundah začne napaka, torej razlika med izračunano razdaljo r in dejansko razdaljo x, hitro povečevati. Pozoren bralec je lahko zasledil, da je pomembna tudi smer, v katero se s telefonom gibljemo. Ta podatek, v kateri smeri se dejansko gibljemo, prav tako izračunamo na podlagi senzorski podatkov, ki vsebujejo šum, zato je rezultat še toliko slabši, ker podatke senzorjev obravnavamo posebej. Vendar pa se lahko s posebnim postopkom, ki se imenuje senzorsko zlivanje, vpliva šuma vseh uporabljenih senzorjev pri izračunu posamezne količine (razdalja, hitrost) do določene mere znebimo, ker podatke senzorjev obravnavamo skupaj. Poglejmo sedaj, kako to dosežemo. To, kar želimo doseči je, za razliko od zgornjega grafa, da se razdalja, po tem, ko se ustavimo, ustali na določeni vrednosti. Prav tako v primeru hitrosti želimo, da se ob ustavitvi hitrost zmanjša na 0.

Knjižnica [6], ki sem jo v tej diplomski nalogi vključil v program na mobilnem telefonu, deluje po principu spodnje sheme (Slika 16).



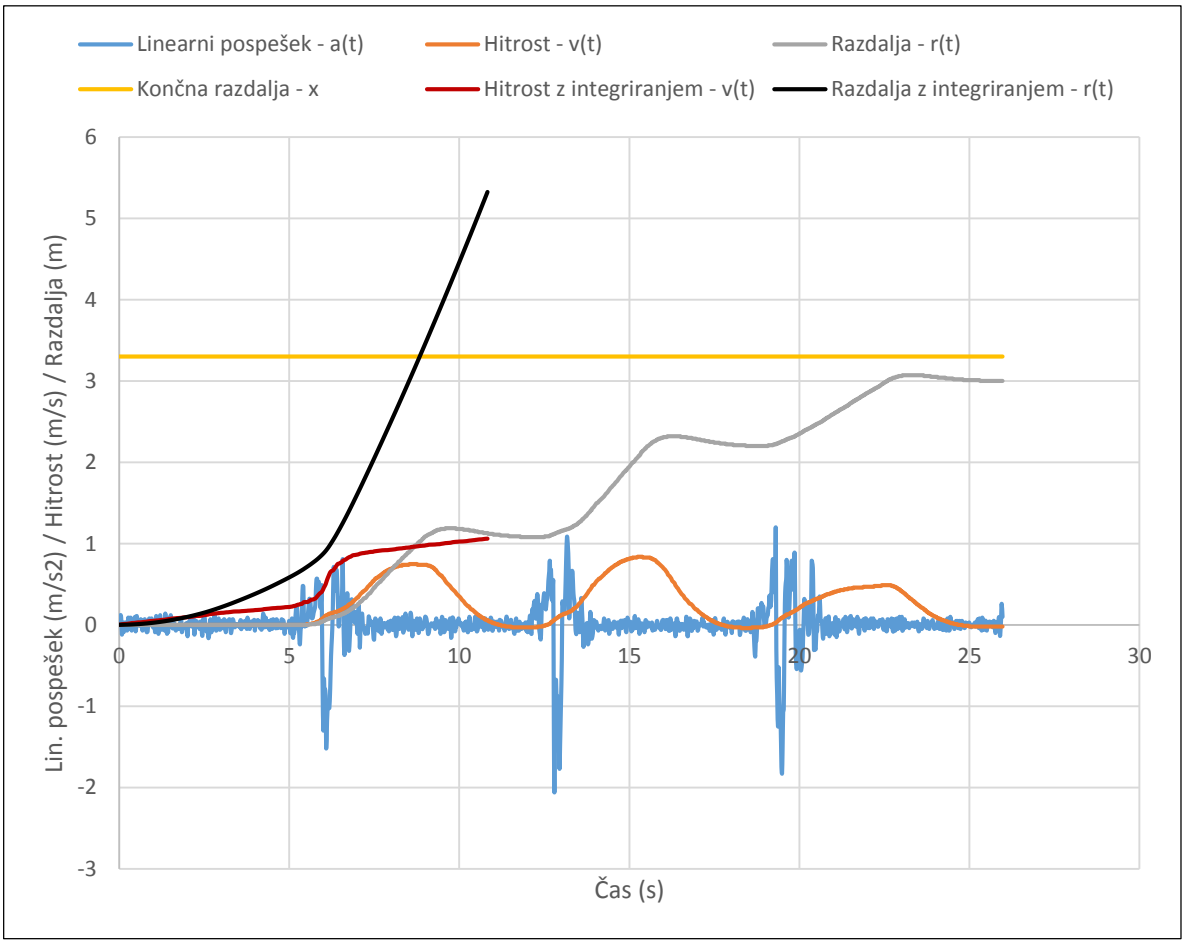
Slika 16: Senzorsko zlivanje

Celoten postopek lahko opišemo v sledečih točkah:

1. Izločitev gravitacijskega pospeška iz pospeškomera.
2. Izračun smeri iz podatkov pospeškomera in magnetometra (smer je tukaj zelo nestabilna).
3. Izračun smeri s senzorskim zlivanjem v 2. točki izračunane smeri in senzorskih podatkov žiroskopa (smer tukaj postane stabilna).
4. Izračun linearnega pospeška za določeno smer. (Če se ne gibljemo po začrtani smeri, se linearni pospešek v času dt množi s faktorjem 1 – odmik / 90°. Če je odmik večji kot 90°, pospešek postane enak 0. Velja za odmik tako v levo kot desno.)
5. Izvajanje algoritma Kalmanov filter, s katerim izračunamo oceno najprej za hitrost, nato pa še oceno za razdaljo.

V spodnjem grafu (Graf 2) vidimo, da sta že po 10 sekundah obe količini pridobljeni z integriranjem (hitrost in razdalja) bistveno odstopali od dejanskih vrednosti, zato ju po tem času ne prikazujemo več. Drugače je pri količinah, pridobljenih s senzorskim zlivanjem. Vidimo, da se tako hitrost kot razdalja lepo prilagodita dejanskim vrednostim. Iz grafa je lepo razviden potek meritve. Najprej smo 5 sekund stali na mestu, nakar smo se premaknili za 1 m naprej in popolnoma ustavili, nato smo se zopet premaknili za 1 m naprej in spet ustavili in spet premaknili za 1 m naprej in dokončno ustavili. Razdalja r, ki jo razberemo na mobilnem

telefonu, se ustavi pri cca. 3 m, kar je 0,3 m razlike od dejanske razdalje, ki smo jo izmerili z laserskim merilnikom.



Graf 2: Bistveno izboljšan rezultat z uporabo senzorskega zlivanja

3.4 Dijkstrov algoritem

Dijkstrov algoritem uporabljamo za iskanje najkrajše poti v usmerjenem in uteženem grafu. Podatek o dolžini, ki je nenegativna, najdemo na posamezni povezavi med dvema točkama, kar lahko imenujemo tudi utež. Algoritem najde minimalno vsoto razdalj med začetno in končno točko in kot rezultat vrne tudi najkrajšo oz. najkrajše poti, če jih je več. V našem primeru uporabljamo samo eno najkrajšo pot. Delovanje algoritma lahko opišemo v psevdokodi (Slika 17).

```
 1  function Dijkstra(Graph, source):
 2      dist[source]  := 0
 3      for each vertex v in Graph:
 4          if v ≠ source
 5              dist[v]  := infinity
 6              previous[v]  := undefined
 7          end if
 8          add v to Q
 9      end for
10
11      while Q is not empty:
12          u := vertex in Q with min dist[u]
13          remove u from Q
14
15          for each neighbor v of u:
16              alt := dist[u] + length(u, v)
17              if alt < dist[v]:
18                  dist[v]  := alt
19                  previous[v]  := u
20              end if
21          end for
22      end while
23      return dist[], previous[]
24  end function
```

Slika 17: Dijkstrov algoritem v psevdokodi [6]

1. Inicializacija:
    a. Nastavi razdalje *dist[v]* = ∞ in prednike *previous[v]* = null za vsako točko *v* (ki ni začetna) ter razdaljo začetne točke *dist[source]* = 0 (vrstice od 2 do 7).
    b. Doda vse točke grafa G v prioritetno vrsto Q (vrstica 8).
2. Izvajanje glavne zanke, dokler prioritetna vrst Q ni prazna:
    a. Poiščemo v vrsti Q točko *u* z najmanjšo razdaljo *dist[u]* oz. prioriteto. Na začetku je to začetna točka (vrstica 12).
    b. Izločimo točko *u* iz prioritetne vrste Q. Če iščemo najkrajšo pot med podano začetno in končno točko, potem tukaj dodamo pogoj *if u = target* in če je ta pogoj izpolnjen, končamo zanko (vrstica 13).
3. Izvajanje notranje zanke, dokler točka *u* nima več sosednjih točk *v* (kjer točka *v* še ni odstranjena iz vrste Q):
    a. Spremenljivki *alt* dodelimo vrednost, ki je enaka vsoti *dist[u]* (vsebuje najmanjšo razdaljo za točko *u*) in *length(u, v),* ki vrne razdaljo med točko *u* in sosednjo točko *v* (vrstica 16).
    b. Spremenljivka *alt* tako vsebuje celotno razdaljo od začetne točke do sosednje točke *v*. Če je ta razdalja manjša kot pa razdalja *dist[v]* za točko *v*, potem *dist[v]* dobi vrednost spremenljivke *alt* in sosednji točki *v* dodelimo tudi prednika, torej pravkar obiskano točko *u,* sicer nadaljujemo z izvajanjem notranje zanke (vrstice od 17 do 20).
4. Po koncu izvajanja glavne zanke preverimo, če je vrednost *previous[target]* definirana, sicer pot od začetne do končne točke ne obstaja. V *dist[target]* (če ni vrednost ∞) preberemo tudi dolžino najkrajše poti. Če se sprehodimo nazaj po seznamu prednikov *previous,* dobimo tudi vse točke na najkrajši poti od končne to začetne točke.

4 Razvoj sistema

4.1 Strojna oprema

Sistem za pozicioniranje je bil razvit na telefonu Samsung Galaxy S3 mini. Telefon vsebuje dvojedrni procesor 1 GHz ARM Cortex A2. Velikost pomnilnika je 1 GB RAM. Povezuje se lahko na brezžično omrežje 802.11 a/b/g/n (maksimalen prenos podatkov do 54 Mbps) ali preko GSM 2G in 3G omrežja (maksimalen prenos podatkov do 14.4 Mbps). Prav tako vsebuje NFC čip (NXP PN547), ki zna komunicirati z ostalimi NFC napravami. Telefon vsebuje različne senzorje, kot so pospeškomer, magnetometer in žiroskop.

4.2 Pospeškomer

Pospeškomer je senzor, ki zna izmeriti pospešek (m/s^2) telefona v vseh 3 smereh koordinatnega sistema (x, y, z). Pospešek se izračuna po 2. Newtonovem zakonu, torej vsota vseh sil, ki delujejo na telefon, deljena z maso telefona. Kot smo že omenili, na izmerjeni pospešek vpliva tudi sila teže, zato moramo v našem primeru ta vpliv izločiti iz meritev. To dosežemo s preprostim filtrom. Spremenljivka α definira, koliko gravitacije je potrebno izločiti iz trenutnega odčitka po spodnji formuli, kjer je t časovna konstanta 100 ms (groba ocena, ki je bila izbrana na podlagi meritev, ki so dale najboljši rezultat) in dt (pretečeni čas v sekundah med posameznimi odčitki, v povprečju je to ca. 10 ms) [7].

$$\alpha = \frac{t}{(t + dt)} \tag{10}$$

Izločitev gravitacije iz pospeška:

$$\begin{aligned} gravX &= \alpha * gravX + (1 - \alpha) * odčitekPospeškaX \\ gravY &= \alpha * gravY + (1 - \alpha) * odčitekPospeškaY \\ gravZ &= \alpha * gravZ + (1 - \alpha) * odčitekPospeškaZ \end{aligned} \tag{11}$$

Izračun linearnega pospeška za vsako smer x, y, z:

$$linearniPospešekX = odčitekPospeškaX - gravX$$
$$linearniPospešekY = odčitekPospeškaY - gravY$$
$$linearniPospešekZ = odčitekPospeškaZ - gravZ$$

(12)

4.3 Magnetometer

Magnetometer oz. senzor magnetnega polja v telefonu meri magnetno polje okoli telefona. Podatki, ki jih pridobimo, so prav tako za vsako smer koordinatnega sistema (x, y, z), posamezni podatki so v enotah µT. V našem primeru so to vhodni podatki skupaj s podatki pospeškomera pri izračunu rotacijske matrike, ki jo dobimo iz vgrajene android knjižnice (*android.hardware.SensorManager*) z uporabo funkcije *getRotationMatrix*. Izsek iz programa je spodaj:

```
private void calcAccMagOrientation() {
    if (SensorManager.getRotationMatrix(rotationMatrix, null, accel, magnet)) {
        SensorManager.getOrientation(rotationMatrix, accMagOrientation);
    }
}
```

Koda 1: Izračun rotacijske matrike, iz katere izračunamo orientacijo telefona.

Kot vidimo iz programske kode, se rotacijska matrika nato uporablja pri izračunu orientacije telefona. Izračunani vektor *accMagOrientation* (vrtenje okoli x, y in z osi v radianih) nastopa v nadaljevanju senzorskega zlivanja (Slika 16) skupaj s podatki iz žiroskopa.

4.4 Žiroskop

Žiroskop meri rotacijo telefona v enotah rad/s za vsako od treh smeri koordinatnega sistema (x, y, z).

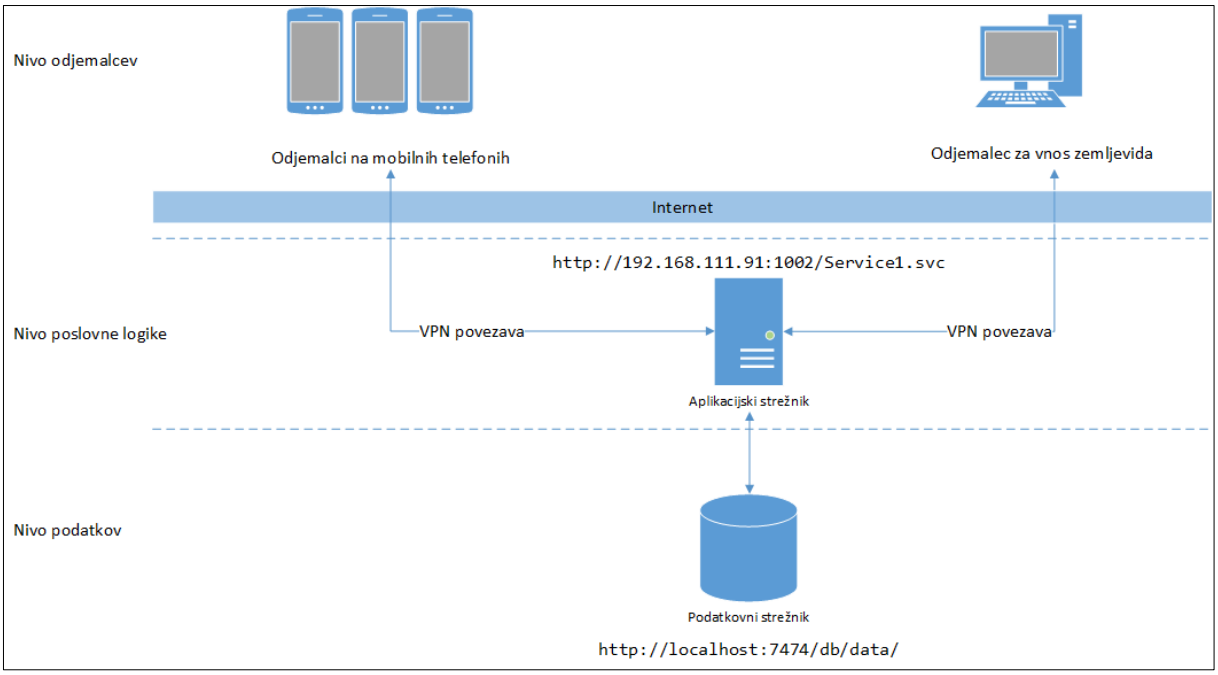
4.5 Programska oprema

Operacijski sistem (OS) mobilnega telefona je Android 4.1 Jelly Bean, na katerem je nameščena aplikacija, ki usmerja uporabnika od začetne do izbrane končne točke. Telefon se preko interneta povezuje na strežnik, kjer je nameščen OS Windows 7 s spletnim strežnikom IIS 7, ki ponuja spletne storitve vsem odjemalcem, torej tudi mobilnim telefonom. Spletne storitve se na strežniku povezujejo na grafno podatkovno bazo Neo4J, kjer je shranjena definicija grafov oz. zemljevidov stavb. V okviru tega diplomskega dela je bila narejena tudi enostavna samostojna aplikacija za vnos in urejanje posameznih lastnosti zemljevida v grafno podatkovno bazo.

Android aplikacija (odjemalec) je bila razvita v razvojnem okolju Eclipse, ki vsebuje dodatek ADT (angl. Android Development Tools) v programskem jeziku JAVA, medtem ko je strežniški del in aplikacija za vnos zemljevida nastala v razvojnem okolju .Net 2012, v programskem jeziku Visual Basic.

4.6 Arhitektura sistema

Spodnja slika (Slika 18) opisuje arhitekturo celotnega sistema. Gre za 3-tirno aplikacijo odjemalec-strežnik.



Slika 18: Arhitektura sistema

Komunikacija med aplikacijskim strežnikom in posameznim odjemalcem na mobilnem telefonu poteka v JSON obliki, kar pomeni, da se vsi objekti na strani aplikacijskega strežnika serializirajo v to obliko in pošljejo na mobilni telefon, kjer se deserializirajo nazaj v objekte. Te objekte v obliki podatkovnih struktur aplikacija na mobilnem telefonu uporablja znotraj svojih procesov. Spodaj so navedeni klici mobilnega odjemalca v obliki URL naslova in njihov odgovor aplikacijskega strežnika v obliki JSON.

http://192.168.111.91:1002/Service1.svc/getPoint?idPoint=4
`{"Floor":1,"Id":4,"Map":1,"Name":"Point D","Type":"stairs","X":1,"Y":1}`

Tabela 3: Poizvedba posamezne točke in rezultat

http://192.168.111.91:1002/Service1.svc/getAllPoints

```
[{"Floor":0,"Id":1,"Map":1,"Name":"Point
A","Type":"normal","X":1,"Y":1},{"Floor":0,"Id":2,"Map":1,"Na
me":"Point
B","Type":"end","X":1,"Y":1},{"Floor":0,"Id":3,"Map":1,"Name"
:"Point
C","Type":"stairs","X":1,"Y":1},{"Floor":1,"Id":4,"Map":1,"Na
me":"Point
D","Type":"stairs","X":1,"Y":1},{"Floor":1,"Id":5,"Map":1,"Na
me":"Point E","Type":"normal","X":1,"Y":1}]
```


Tabela 4: Poizvedba vseh točk v grafu in rezultat

http://192.168.111.91:1002/Service1.svc/GetAllLeafPoints
`[{"Floor":1,"Id":5,"Map":1,"Name":"Point E","Type":"normal","X":1,"Y":1}]`

Tabela 5: Poizvedba po vseh končnih točkah (listih) in rezultat

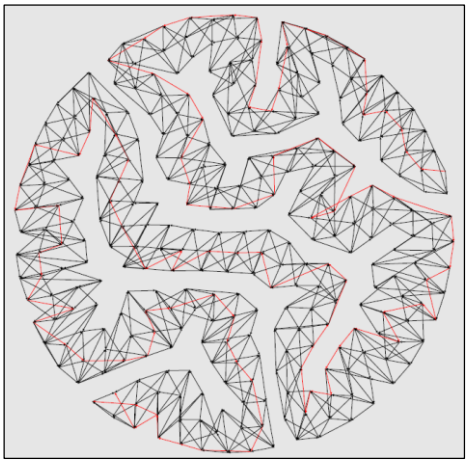
http://192.168.111.91:1002/Service1.svc/getShortestPath?idStartPoint=1&idEndPoint=5
`{"EndPoint":{"Floor":1,"Id":5,"Map":1,"Name":"Point E","Type":"normal","X":1,"Y":1},"Relations":[{"Direction":330,"Distance":1.0,"Heading":"Down the hall.","IsForwared":false,"PointFrom":{"Floor":0,"Id":1,"Map":1,"Name":"Point A","Type":"normal","X":1,"Y":1},"PointTo":{"Floor":1,"Id":4,"Map":1,"Name":"Point D","Type":"stairs","X":1,"Y":1}},{"Direction":30,"Distance":1.0,"Heading":"Down the hall.","IsForwared":false,"PointFrom":{"Floor":1,"Id":4,"Map":1,"Name":"Point D","Type":"stairs","X":1,"Y":1},"PointTo":{"Floor":1,"Id":5,"Map":1,"Name":"Point E","Type":"normal","X":1,"Y":1}}],"StartPoint":{"Floor":0,"Id":1,"Map":1,"Name":"Point A","Type":"normal","X":1,"Y":1},"Steps":2,"TotalDistance":2.0}`

Tabela 6: Poizvedba po najkrajši poti in rezultat

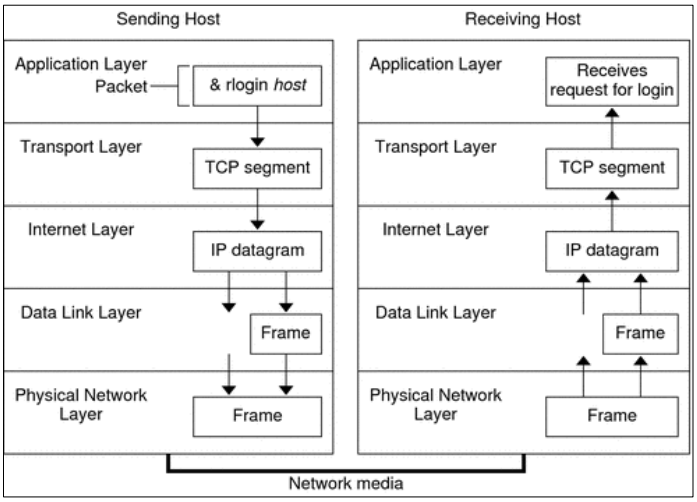
Rezultati meritev poizvedb skozi omrežje pri iskanju najkrajše poti od začetne do končne točke za različno število generiranih točk (Tabela 7).

Št. generiranih točk	Količina poslanih podatkov (kB)	Čas za poizvedbo (s)
300	24	0,8
600	48	1,5
1800	144	4,1
3600	288	8,7

Tabela 7: Rezultati meritev



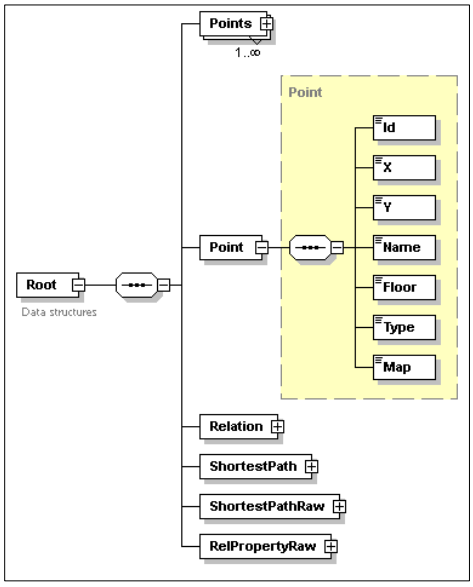
Slika 19: Zemljevid s 300 generiranimi točkami



Slika 20: TCP/IP sklad prikazuje potovanje poizvedbe preko omrežja.

4.7 Podatkovne strukture

Zgoraj smo opisali postopek, kako se pretvarjajo objekti v JSON in obratno in kako poteka prenos skozi omrežje, ničesar pa še nismo povedali o vsebini posameznih objektov in kje posamezne elemente objektov uporabljamo. V okviru te diplomske naloge je bila razvita XML shema vseh podatkovnih struktur, kar prikazujejo tudi spodnje slike. S posebnimi generatorji programske kode lahko XML sheme pretvorimo direktno v podatkovne strukture ciljnega programskega jezika. V našem primeru smo generirali shemo XML tako v objekte programskega jezika JAVA, kot tudi objekte programskega jezika Visual Basic. S tem smo ohranili enako strukturo podatkov tako na strani odjemalca kot tudi na strani strežnika. Za v bodoče pa nam XML shema omogoča razvoj podatkovnih struktur še za kakšen drug programski jezik, npr. C++ ali Objective-C, če se odločimo za razvoj aplikacije za iPhone operacijskega sistema iOS.



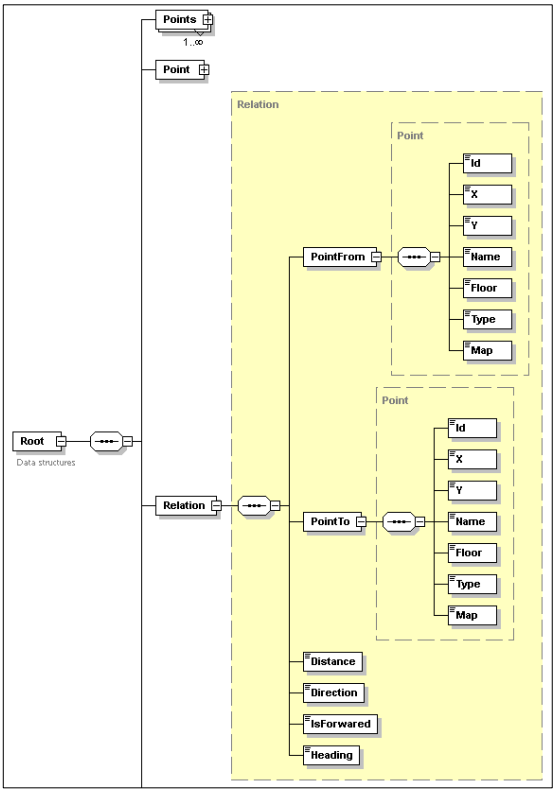
Slika 21: Podatkovna struktura Point

Podatkovna struktura Point opisuje objekt točke s posameznimi lastnostmi, kot so Id, X, Y, Name, Floor, Type in Map. Posamezen pomen opisuje spodnja tabela.

Lastnost	Tip podatka	Pomen
Id	int	Enolični identifikator NFC nalepke oz. točke v grafu
X	decimal	Koordinata X
Y	decimal	Koordinata Y
Name	string	Ime točke
Floor	int	Številka nadstropja

Type	string	Tip točke (npr. začetna, stopniščna, končna, vmesna)
Map	int	Številka zemljevida, kamor ta točka spada

Tabela 8: Opis podatkovne strukture Point

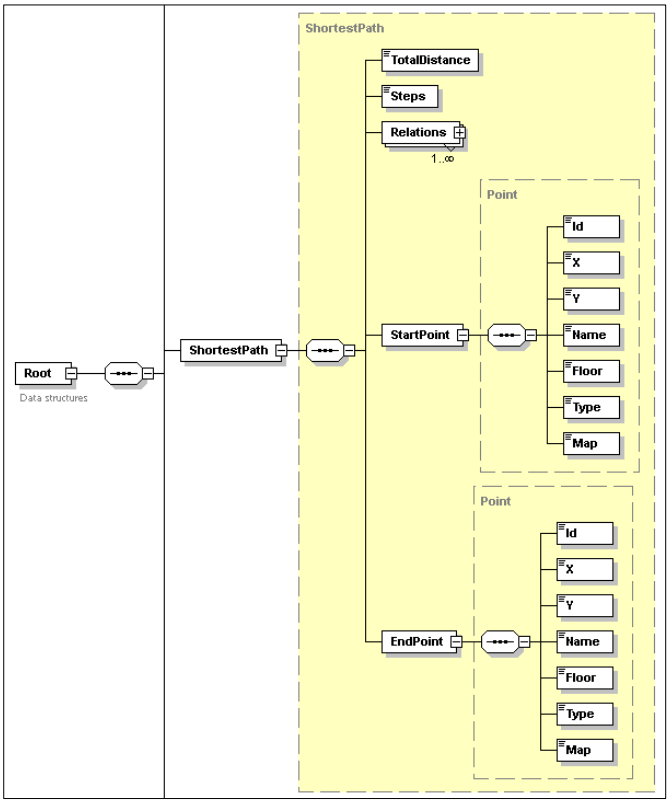


Slika 22: Podatkovna struktura Relation

Podatkovna struktura Relation opisuje objekt povezave med dvema točkama. Opis posameznih elementov je v spodnji tabeli.

Lastnost	Tip podatka	Pomen
PointFrom	Point	Točka, iz katere izhajamo
PointTo	Point	Točka, v katero smo namenjeni
Distance	decimal	Razdalja med dvema točkama v metrih
Direction	decimal	Smer v stopinjah
IsForwared	boolean	Ali je povezava enosmerna
Heading	string	Tekstovni opis povezave (npr. ob stopnišču, pri jedilnici)

Tabela 9: Opis podatkovne strukture Relation



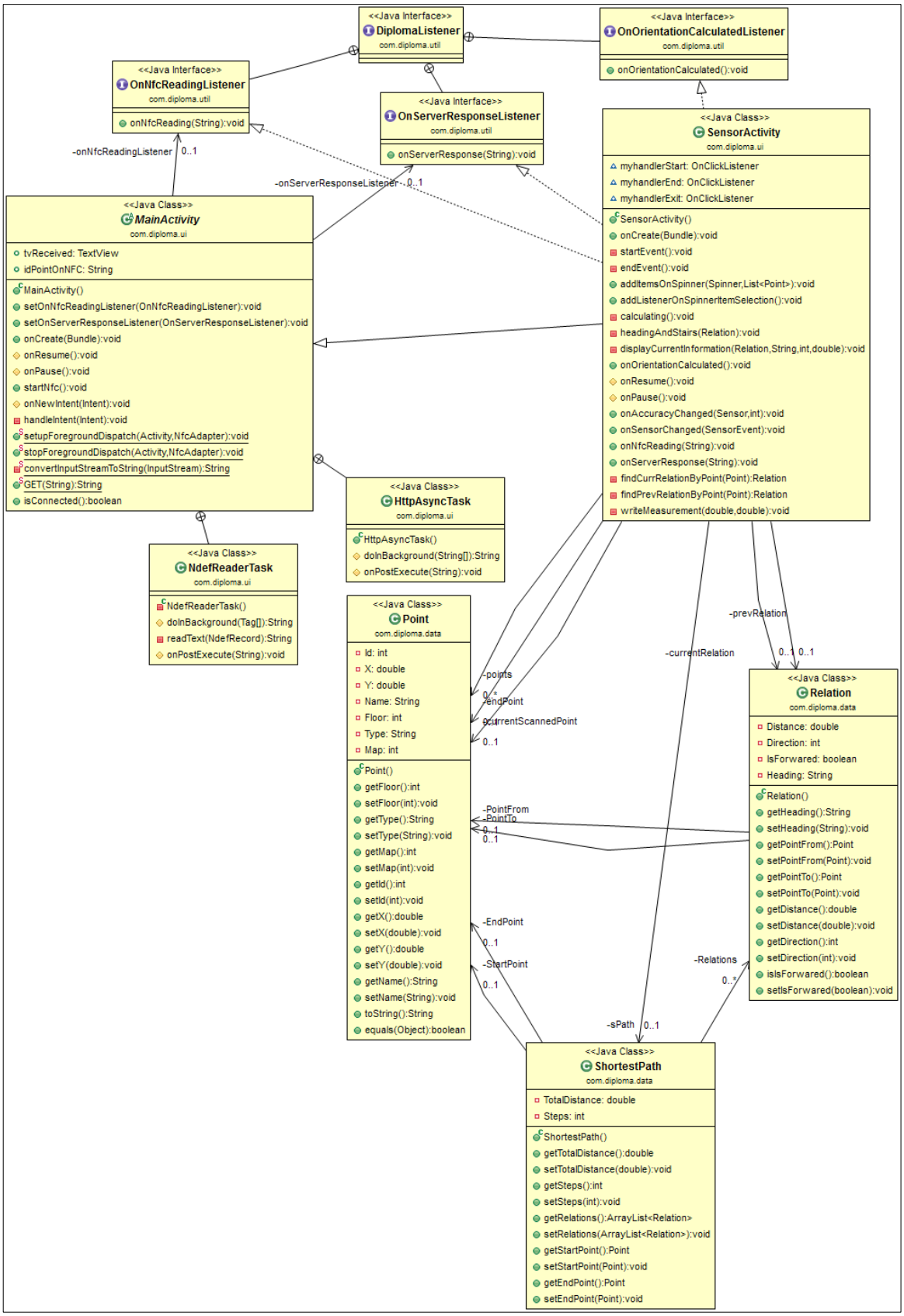
Slika 23: Podatkovna struktura ShortestPath

Podatkovna struktura ShortestPath opisuje najkrajšo pot, ki jo izračuna algoritem Dijkstra. Opis posameznih elementov je v spodnji tabeli.

Lastnost	Tip podatka	Pomen
TotalDistance	decimal	Celotna razdalja v metrih od začetne do končne točke
Steps	int	Število vseh povezav na poti
Relations	List (Of Relation)	Seznam tistih povezav, ki so na poti
StartPoint	Point	Začetna točka
EndPoint	Point	Končna točka

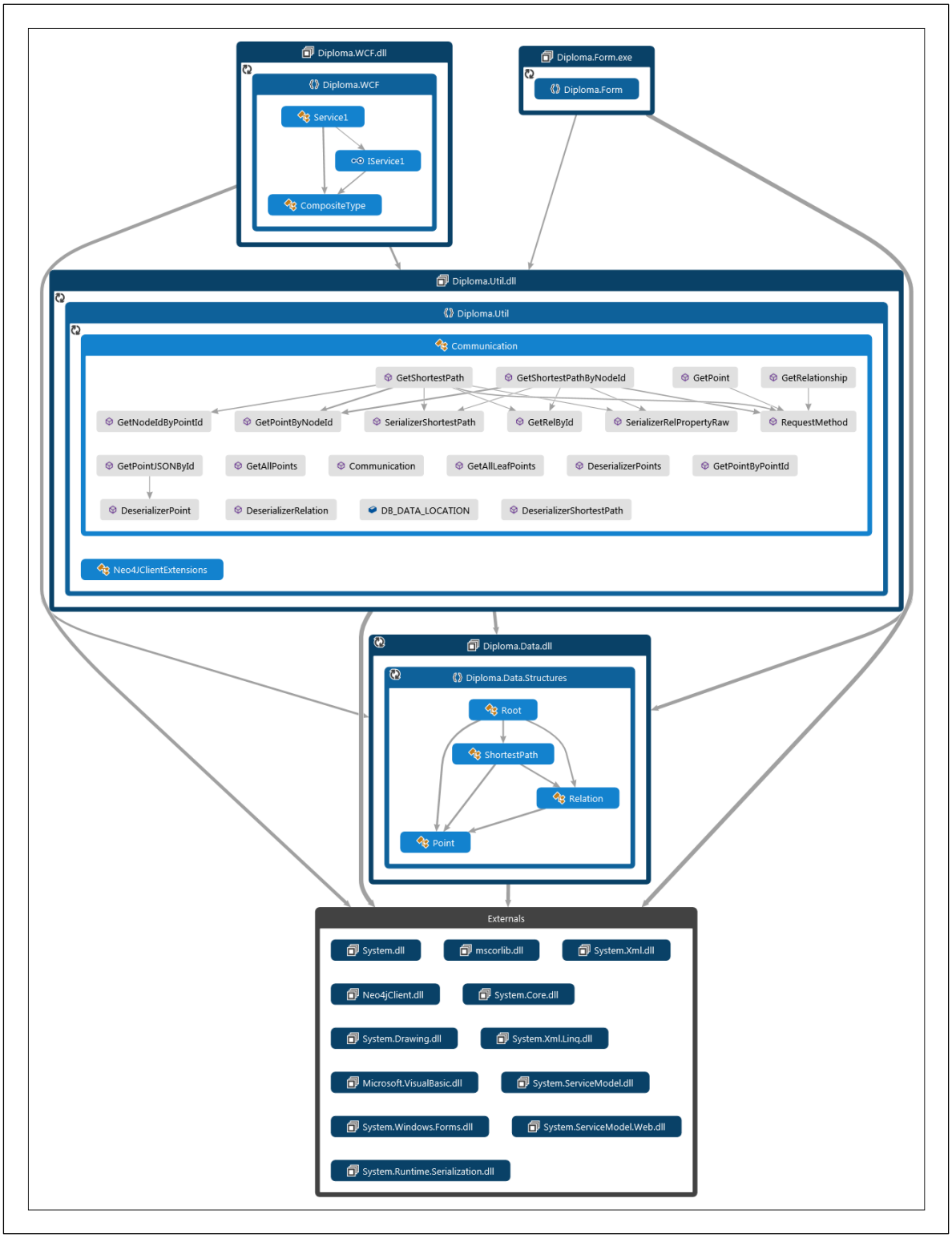
Tabela 10: Opis podatkovne strukture ShortestPath

4.8 Mobilna aplikacija



Slika 24: UML razredni diagram mobilne aplikacije

4.9 Strežniška aplikacija



Slika 25: Graf odvisnosti med strežniškimi moduli

4.10 Podatkovna baza

Podatkovna baza, ki jo v našem sistemu uporabljamo, je grafna podatkovna baza Neo4j, kjer za razliko od relacijskih podatkovnih baz, ki so danes najbolj razširjene, ne uporabljamo tabel, temveč matematične grafe kot posamezne gradnike. Problem pri relacijskih podatkovnih bazah je predvsem v primerih, ko imamo zelo povezane podatke in so zato poizvedbe, ki vsebujejo veliko stikov (angl. joins) med tabelami, zelo počasne in lahko trajajo zelo dolgo, kar pa si v našem primeru ne moremo privoščiti, saj potrebujemo podatke v realnem času. V primeru grafnih podatkovnih baz, ko podatki hitro naraščajo in so med seboj zelo povezani, je učinkovitost in hitrost poizvedb dokaj konstantna [8]. Razlog se skriva v preiskovanju lokaliziranega dela grafa. Čas posamezne poizvedbe je sorazmeren z velikostjo preiskovanja tega dela, ne pa celotnega grafa. Namen tega dela ni primerjava grafne in relacijske podatkovne baze, ampak samo kot ilustracijo razlike v zmogljivosti navedimo primer eksperimenta, ki ga knjiga [8] navaja. Spodnja tabela prikazuje čase izvajanja poizvedbe relacijske in grafne podatkovne baze za primer poizvedbe v socialnem omrežju, ki išče v množici z milijon osebami (vsaka oseba ima v povprečju 50 prijateljev) vse prijatelje prijateljev (t. i. oddaljenih prijateljev) do določene globine (2, 3, 4, 5). Torej globina 2 pomeni Oseba A – Oseba B – Oseba C, globina 3 pomeni Oseba A – Oseba B – Oseba C – Oseba D itd. (znak – pomeni relacijo »je prijatelj«).

Globina	Čas (s) relacijska podatkovna baza	Čas (s) grafna podatkovna baza	Število vrnjenih zapisov
2	0,016	0,010	~2500
3	30,267	0,168	~110.000
4	1543,505	1,359	~600.000
5	nedokončano	2,132	~800.000

Tabela 11: Čas izvajanja poizvedbe pri relacijski in grafni podatkovni bazi

Kot vidimo iz zgornje tabele, je razlika več kot očitna. V našem primeru lahko z grafno podatkovno bazo dokaj enostavno iščemo najkrajše poti določenega dela grafa, ki lahko vsebuje ogromno število točk in povezav, in kar je najpomembneje, da so poizvedbe hitre. Predstavljajmo si ogromno stavbo (npr. muzej Louvre v Parizu), kjer je razporejenih milijon NFC nalepk, mi pa želimo priti pet korakov (povezav) naprej po najkrajši poti. V tem primeru ne bi preiskovali vseh milijon točk, ampak samo tiste, ki so od trenutne točke oddaljene 5 korakov.

Sedaj pa si poglejmo, kako komuniciramo z grafno podatkovno bazo. Za razliko od relacijske podatkovne baze, kjer uporabljamo SQL (angl. Structured query language), tukaj uporabljamo NoSQL (angl. not only SQL). V spodnji programski kodi lahko vidimo, kako v programskem jeziku Visual Basic tvorimo razne poizvedbe. Neo4j pozna več načinov za tvorjenje poizvedb [9], v našem primeru uporabljamo t. i. povpraševalni jezik Cypher za enostavne poizvedbe in t.

i. REST API za poizvedbe po najkrajši poti s standardnimi metodami HTTP (angl. HTTP Request), predvsem z metodo POST.

```
17      Public Function GetPointByPointId(ByVal id As Integer) As Point
18          Dim client = New GraphClient(New Uri(DB_DATA_LOCATION))
19          client.Connect()
20
21          Dim query = client.Cypher.Start(New With {Key .p = All.Nodes}) _
22                      .Where(Function(p As Point) p.Id = id) _
23                      .Return(Of Point)("p")
24
25          Return query.Results.ToList(0)
26      End Function
```

Koda 2: Poizvedba posamezne točke po Id s Cypher poizvedbo

```
66      Public Function GetAllPoints() As List(Of Point)
67          Dim client = New GraphClient(New Uri(DB_DATA_LOCATION))
68          client.Connect()
69          Dim query = client.Cypher.Start(New With {Key .n = All.Nodes}) _
70                              .Return(Of Point)("n") _
71                              .OrderBy("n.Name")
72
73          Return query.Results
74      End Function
```

Koda 3: Poizvedba po vseh točkah s Cypher poizvedbo

```
76      Public Function GetAllLeafPoints() As List(Of Point)
77          Dim client = New GraphClient(New Uri(DB_DATA_LOCATION))
78          client.Connect()
79          Dim query = client.Cypher.Start(New With {Key .n = All.Nodes}) _
80                              .Match("n-[r*]->m") _
81                              .Where("not(m-->())") _
82                              .ReturnDistinct(Of Point)("m") _
83                              .OrderBy("m.Name")
84
85          Return query.Results
86      End Function
```

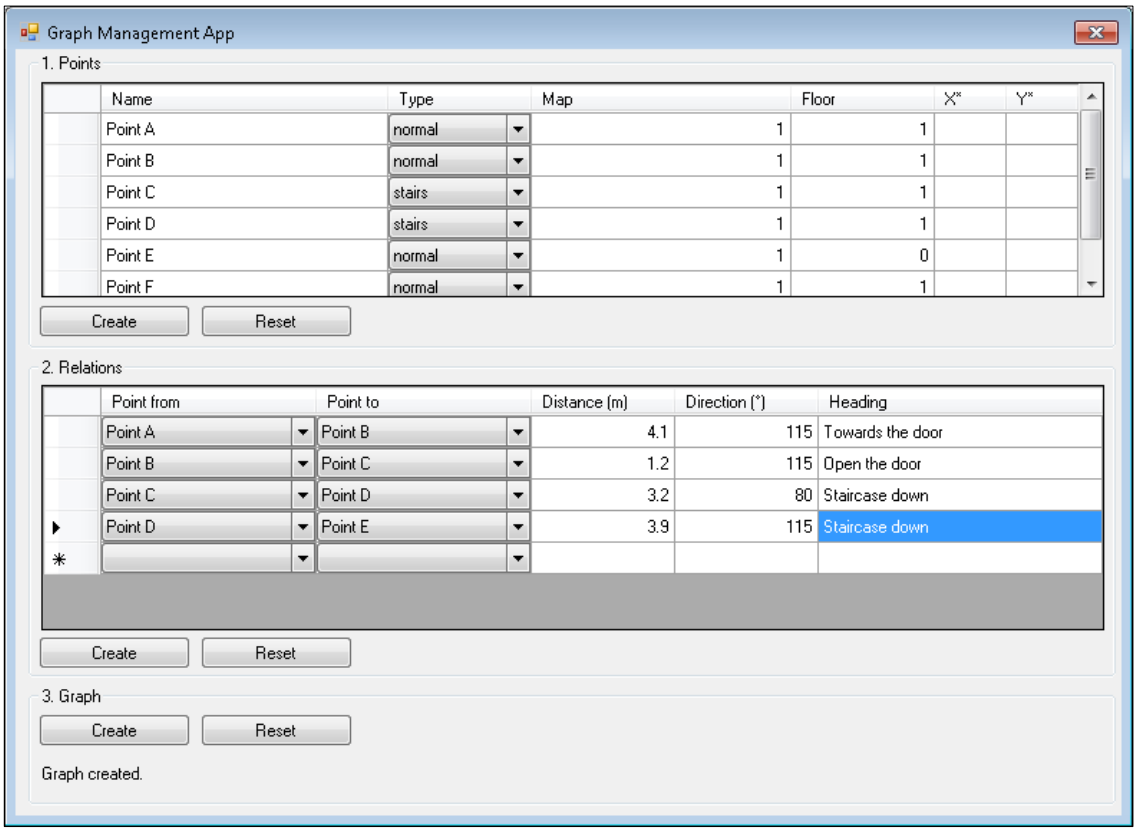
Koda 4: Poizvedba po vseh končnih točkah (listih)

```
142     Public Function GetShortestPathByNodeId(ByVal idStartPoint As Integer, ByVal idEndPoint As Integer) As ShortestPath
143         Dim client = New GraphClient(New Uri(DB_DATA_LOCATION))
144         client.Connect()
145
146         Dim idStartNode = idStartPoint
147         Dim idEndNode = idEndPoint
148
149         Dim json As String = "{" & vbCr & vbLf & _
150            "                          'to' : 'http://localhost:7474/db/data/node/@endNode@'," & vbCr & vbLf & _
151            "                          'cost_property' : 'Cost'," & vbCr & vbLf & _
152            "                          'relationships' : {" & vbCr & vbLf & _
153            "                                            'type' : 'do'," & vbCr & vbLf & _
154            "                                            'direction' : 'out'" & vbCr & vbLf & _
155            "                                          }," & vbCr & vbLf & _
156            "                           'algorithm' : 'dijkstra'" & vbCr & vbLf & _
157            "                }"
158         json = Replace(json, "@endNode@", idEndNode.ToString)
159         Dim js As String = RequestMethod(New Uri(DB_DATA_LOCATION & "/node/" & idStartNode & "/path"), Replace(json, "'"c, """c), "POST")
160         If (js = "") Then
161             Return Nothing
162         End If
163
164         Dim sPathRaw = SerializerShortestPath(js)
```

Koda 5: Poizvedba po najkrajši poti s POST (angl. HTTP Request)

4.11 Primer aplikacije za vnos zemljevida

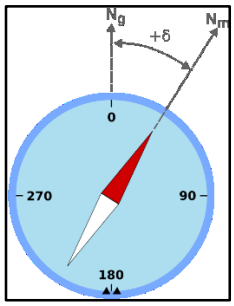
Kot smo že prej omenili, je v okviru te diplomske naloge bila izdelana tudi samostojna aplikacija za enostavni vnos zemljevida (Slika 26) z vsemi pripadajočimi lastnostmi, ki jih potrebujemo v aplikaciji na mobilnem telefonu.



Slika 26: Aplikacija za vnos enostavnega zemljevida v grafno podatkovno bazo

5 Analiza vpliva magnetnega polja

Pri samem testiranju celotnega sistema smo občasno naleteli na izrazito odstopanje pri meritvi magnetnega polja v okolici telefona. Pojav odstopanja sicer ni bil dokaj pogost, pa vendar zaradi napačnega usmerjanja uporabnika zasluži posebno analizo, saj se želimo v celoti izogniti takšnim situacijam. Na podlagi te analize je bil izdelan poseben dodatek v samem programu na mobilnem telefonu, ki uporabnika sproti obvesti, da se nahaja v območju nenormalnega vpliva magnetnega polja, in svetuje, da se uporabnik skupaj s telefonom premakne iz območja. Ta območja so v bližini raznih železnih konstrukcij, drugih elektronskih naprav, samega magneta in ostalih stvari, ki s svojim magnetnim poljem vplivajo na meritev jakosti magnetnega polja, ki se v povprečju giblje od 25 do 65 µT [10]. Znano je, da se magnetno polje Zemlje neprestano spreminja. Presenetljiv je tudi podatek, da se npr. severni pol Zemlje premakne za okoli 10 km na leto. Znane so tudi celo dnevne spremembe lege magnetnega polja, ki so v glavni meri posledice vpliva Sonca. Sonce neprenehoma bruha v medplanetarni prostor nabite delce (sončni veter), ki zmotijo magnetno polje Zemlje, ko zadenejo ob njeno magnetosfero. Torej, tudi če vzamemo v roke kompas, magnetna igla ne kaže proti geografskemu severnemu tečaju (kjer os vrtenja Zemlje okoli lastne osi seka površino Zemlje), pač pa proti magnetnemu severnemu polu (kjer so silnice magnetnega polja navpične glede na zemeljsko površino) [11]. Kot med obema smerema dosega na področju Slovenije okoli 1,5°, za npr. New York znaša 13° (deklinacija).



Slika 27: Deklinacija – kot med geografskim (Ng) in magnetnim (Nm) severom

Vpliv deklinacije lahko v našem primeru zaenkrat ignoriramo, vpliv močnega magnetnega polja pa ne, saj lahko le-ta uporabnika usmeri tudi za 180°, zato je potrebno imeti mehanizem, ki to tudi ugotovi in obvesti uporabnika.

5.1 Meritve

Izračun magnetnega polja iz podatkov magnetometra za vsako smer koordinatnega sistema (x, y, z) poteka po spodnji formuli:

$$M_z = \sqrt{x^2 + y^2 + z^2}$$

(13)

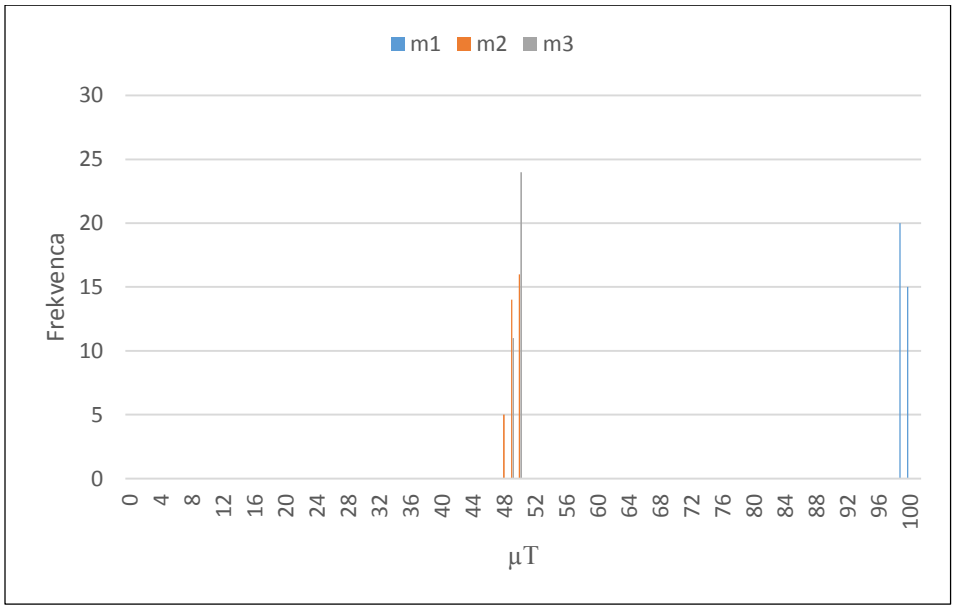
Izmerjeni so bili trije vzorci velikosti n=35 (kot prikazuje tabela spodaj). Predpostavimo, da je vsaka meritev v posameznem vzorcu pridobljena medsebojno neodvisno in naključno, saj nimamo posebnega razloga, da bi vplivali na medsebojne meritve.

N	m1	m2	m3	m1>m2	m1>m3	m2>m3	m1-m2	m1-m3	m2-m3
1	99,04	47,51	49,61	1	1	0	51,53	49,43	-2,10
2	98,88	47,47	49,57	1	1	0	51,41	49,31	-2,10
3	98,92	47,65	49,73	1	1	0	51,27	49,19	-2,08
4	98,84	47,81	49,84	1	1	0	51,03	49,00	-2,03
5	98,87	47,93	49,76	1	1	0	50,94	49,11	-1,83
6	98,83	48,08	49,67	1	1	0	50,75	49,16	-1,59
7	98,79	48,27	49,46	1	1	0	50,52	49,33	-1,19
8	98,80	48,56	49,32	1	1	0	50,24	49,48	-0,76
9	98,83	48,57	49,18	1	1	0	50,26	49,65	-0,61
10	98,83	48,48	49,06	1	1	0	50,35	49,77	-0,58
11	98,85	48,36	48,94	1	1	0	50,49	49,91	-0,58
12	98,89	48,49	48,92	1	1	0	50,40	49,97	-0,43
13	98,94	48,65	48,88	1	1	0	50,29	50,06	-0,23
14	98,99	48,92	48,93	1	1	0	50,07	50,06	-0,01
15	99,09	49,10	49,04	1	1	1	49,99	50,05	0,06
16	99,08	49,09	49,13	1	1	0	49,99	49,95	-0,04
17	99,04	49,41	49,19	1	1	1	49,63	49,85	0,22
18	99,07	49,52	49,23	1	1	1	49,55	49,84	0,29
19	99,01	49,61	49,30	1	1	1	49,40	49,71	0,31
20	99,02	49,55	49,25	1	1	1	49,47	49,77	0,30
21	99,01	49,40	49,33	1	1	1	49,61	49,68	0,07
22	99,05	49,07	49,46	1	1	0	49,98	49,59	-0,39
23	99,03	48,99	49,39	1	1	0	50,04	49,64	-0,40
24	99,04	48,96	49,35	1	1	0	50,08	49,69	-0,39
25	99,04	49,00	49,37	1	1	0	50,04	49,67	-0,37

26	99,01	49,02	49,43	1	1	0	49,99	49,58	-0,41
27	99,04	49,00	49,20	1	1	0	50,04	49,84	-0,20
28	98,98	48,95	49,06	1	1	0	50,03	49,92	-0,11
29	99,00	49,04	48,87	1	1	1	49,96	50,13	0,17
30	98,87	49,10	48,55	1	1	1	49,77	50,32	0,55
31	98,89	49,07	48,38	1	1	1	49,82	50,51	0,69
32	98,92	49,19	48,20	1	1	1	49,73	50,72	0,99
33	98,91	49,09	48,07	1	1	1	49,82	50,84	1,02
34	99,00	49,25	48,07	1	1	1	49,75	50,93	1,18
35	99,01	49,22	48,26	1	1	1	49,79	50,75	0,96

Tabela 12: Meritve magnetnega polja

Spodnji graf prikazuje frekvenčno porazdelitev vrednosti vseh treh meritev. Iz grafa je razvidno, da meritev m1 bistveno izstopa po izmerjenih vrednostih. V naslednjem podpoglavju bomo poskušali dokazati, da gre pri meritvi m1 za anomalijo in da lahko v takšnih primerih upravičeno trdimo, da gre za dodaten vpliv močnega magnetnega polja. Uporabnika telefona zato lahko v takšnih primerih pravočasno opozorimo, da trenutno usmerjanje ni točno.



Graf 3: Primerjava med meritvami

5.2 Postopki in izračuni

Pri analizi si lahko pomagamo s testiranjem statistične hipoteze ali domneve. Statistična hipoteza je domneva o porazdelitvi slučajne spremenljivke na populaciji. Verjetnost, da zavrnemo pravilno hipotezo, označujemo z α in je tipično 0 ali 1 ali 0,05 ali 0,01. V našem primeru bomo uporabljali $\alpha = 0,05$.

Postopek je sledeč:

1. Postavimo t. i. ničelno hipotezo H_0.
2. Postavimo alternativno hipoteza H_A, ki je komplementarna hipotezi H_0.
3. Izberemo stopnjo značilnosti testa α.
4. Izberemo ustrezno testno statistiko T.
5. Določimo kritično območje K_α, torej območje možne zavrnitve hipoteze.
6. Izračunamo T na vzorcu.
7. Če je T na vzorcu v kritičnem območju, potem hipotezo H_0 zavrnemo, sicer H_0 ne zavrnemo.

V našem primeru testiramo hipotezo, da je jakost magnetnega polja v Ljubljani v tem času v povprečju približno 48,95μT.

1. Ničelna hipoteza H_0: $\mu_0 = 48{,}95\mu T$
2. Alternativna hipoteza H_A: μ_A ni enako 48,95μT
3. Stopnja značilnosti $\alpha = 0{,}05$
4. Testna statistika

Naslednji problem, s katerim se soočimo, je izbor testne statistike. Ker imamo več kot 30 meritev (n = 35) in ker za slučajno spremenljivko X (kot tudi Y in Z) ne moremo trditi, da je porazdeljena normalno, lahko pa zato trdimo, da je vzorčna statistika V porazdeljena približno standardizirano normalno, se pravi po zakonu N(0, 1). Zato lahko za testno statistiko vzamemo:

$$T = \frac{\bar{X} - \mu_0}{S}\sqrt{n} \tag{14}$$

Pri čemer je popravljena vzorčna deviacija:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{15}$$

In povprečje:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{16}$$

5. Določimo kritično območje:

$$K_\alpha = \left(-\infty, -t_{1-\frac{\alpha}{2}}\right) \bigcup \left(t_{1-\frac{\alpha}{2}}, \infty \right) = \ (-\infty, -1{,}96) \bigcup (\ 1{,}96, \infty)$$

(17)

6. Izračunamo T na vzorcih:

$T_{m2} = \frac{48{,}78 - 48{,}95}{0{,}58} \sqrt{35} = \ -1{,}73$

$T_{m3} = \frac{49{,}11 - 48{,}95}{0{,}58} \sqrt{35} = \ 1{,}63$

(18)

7. Ovrednotimo rezultate:
   T_{m2} vrednost ne pade v kritično območje, hipoteze H_0 ne moremo zavrniti.
   T_{m3} vrednost ne pade v kritično območje, hipoteze H_0 ne moremo zavrniti.

Z naslednjima dvema testoma preverimo, ali so slučajne spremenljivke vseh treh meritev na isti populaciji enako porazdeljene. Slučajno spremenljivko prve meritve označimo z X, druge meritve z Y in tretje z Z. Preverimo hipoteze H_0, ali so:

1. X enako porazdeljena kot Y, označimo $X \sim Y$
2. X enako porazdeljena kot Z, označimo $X \sim Z$
3. Y enako porazdeljena kot Z, označimo $Y \sim Z$

(19)

Prvi test, ki ga izvedemo, je t. i. test z znaki. V tabeli (Tabela 12) določimo stolpce m1 > m2, m1 > m3 in m2 > m3 tako, da ob izpolnjenem pogoju »večje« postavimo vrednost 1 v polje tabele, ob izpolnjenem pogoju »enako« vrednost 0,5, sicer 0, kot prikazuje tabela (Tabela 12) Seštejemo vse vrednosti 1 po posameznih stolpcih. Vsote označimo po vrsti K_1, K_2, K_3. Vsote K so tudi slučajne spremenljivke, ki so ob veljavnosti hipoteze H_0 porazdeljene po binomskem zakonu B(0, 1/2), velja namreč:

$$P(K = k) = \binom{n}{k} 2^{-n}$$

(20)

Postopek enako kot zgoraj (19) izvedemo za vse tri možnosti:

1. Postavimo ničelno hipotezo H_0: $X \sim Y$
2. Postavimo alternativno hipotezo H_A: X ni enako porazdeljena kot Y
3. Izberemo stopnjo značilnosti testa $\alpha = 0{,}05$
4. Izberemo ustrezno testno statistiko:

$$T = \frac{2K - n}{\sqrt{n}}$$

(21)

5. Določimo kritično območje K_α, torej območje možne zavrnitve hipoteze:

$$K_\alpha = \left(-\infty, -t_{1-\frac{\alpha}{2}}\right) \bigcup \left(t_{1-\frac{\alpha}{2}}, \infty\right) = (-\infty, -1{,}96) \bigcup (1{,}96, \infty)$$

(22)

6. Izračunamo T na vzorcih:

   $T_{k3} = \frac{2*13-35}{\sqrt{35}} = -1{,}52$

(23)

7. Ovrednotimo rezultate:

   T_{k3} vrednost ne pade v kritično območje, hipoteze H_0 ne moremo zavrniti, za Y in Z ne moremo trditi, da sta enako porazdeljeni.

Drugi test, ki ga izvedemo, je t. i. test z rangi. V tabeli (Tabela 12) določimo stolpce m1 - m2, m1 - m3 in m2 - m3. Razvrstimo absolutne razlike tako, da preštejemo pozitivne range W^+ in negativne range W^-. Za W izberemo manjšo vrednost:

$$W = min\{W^+, W^-\}$$

(24)

Zopet izvedemo postopek kot zgoraj (19) za vse tri možnosti:

1. Postavimo ničelno hipotezo H_0: $X \sim Y$ torej $W^+ \sim W^-$
2. Postavimo alternativno hipoteza H_A: X ni enako porazdeljena kot Y
3. Izberemo stopnjo značilnosti testa $\alpha = 0{,}05$
4. Izberemo ustrezno testno statistiko:

$$T = \frac{W - \mu_W}{\sigma_W} \sim N(0, 1)$$

(25)

Pri čemer je:

$$\mu_W = E(W) = \frac{1}{4} n(n+1)$$

$$\sigma_W = \sqrt{\frac{1}{24} n(n+1)(2n+1)}$$

(26)

5. Določimo kritično območje K_α, torej območje možne zavrnitve hipoteze:

$$K_\alpha = \left(-\infty, -t_{1-\frac{\alpha}{2}}\right) \bigcup \left(t_{1-\frac{\alpha}{2}}, \infty\right) = (-\infty, -1{,}96) \bigcup (1{,}96, \infty) \quad (27)$$

6. Izračunamo T na vzorcih:

$T_{w3} = \frac{12-315}{61{,}05} = -4{,}96$ (28)

7. Ovrednotimo rezultate:

   T_{w3} vrednost pade v kritično območje, hipotezo H_0 zavrnemo, Y in Z nista enako porazdeljeni.

Kot smo lahko videli, test z znaki nam ne da rezultata o enaki porazdelitvi Y in Z, medtem ko test z rangi to hipotezo zavrne, torej v resnici Y in Z nista enako porazdeljeni. Torej smo pri prvem testu, ko smo testirali, da je jakost magnetnega polja za Ljubljano v povprečju približno 48,95μT, pravilno izbrali testno statistiko, saj slučajne spremenljivke X, Y in Z niso med seboj enako porazdeljene. V našem programu na mobilnem telefonu lahko vključimo preverjanje, ki na podlagi vzorca meritev (n > 30) sproti obvešča uporabnika, ali se nahaja v območju, kjer usmerjanje ne deluje pravilno.

6 Sklepne ugotovitve

Trenutno v splošni uporabi še ne najdemo veliko podobnih sistemov za pozicioniranje v zaprtih prostorih. Razlogi so najverjetneje v tem, da zaenkrat malo ljudi uporablja pametne telefone, v zaračunavanju mobilnih operaterjev po prenosu prenešenih podatkov, v razširjenosti brezplačnih brezžičnih dostopnih točk in dodatnih stroških ponudnika tovrstne storitve, ki zaenkrat v tem še ne vidi dodane vrednosti. Vendar pa se že kažejo začetki, npr. v podjetju Google že objavljajo na svojih zemljevidih natančne tlorise stavb, na to temo je bila napisana že kopica člankov, npr. [12], v katerem vidijo pozicioniranje v zaprtih prostorih kot novo možnost spremljanja navad kupcev ipd. Navedeni razlogi se bodo s časom prav gotovo razblinili, pametni telefoni bodo s časom še bolj razširjeni in zmogljivi, mobilni operaterji bodo prenos podatkov zaračunavali fiksno, ne glede na količino prenešenih podatkov, brezžične dostopne točke bodo skoraj povsod, pa tudi ponudniki te storitve bodo prepoznali nove možnosti za ustvarjanje dodane vrednosti.
V tej diplomski nalogi sem predstavil enega od možnih sistemov pozicioniranja v zaprtih prostorih, ki je dokaj natančen. Natančnost je tukaj sestavljena in jo lahko definiramo:

1. z gostoto razporeditve NFC nalepk in določitev njihovih točnih lokacij (ali določitev oddaljenosti med vsemi pari točk ter njihove smeri),
2. z natančnostjo prikaza smeri do določene NFC nalepke,
3. kot razliko med dejansko in s telefonom izmerjeno oddaljenostjo do določene NFC nalepke.

Prva je odvisna predvsem od tistega, ki pripravi takšen zemljevid, drugi dve pa od prve in od natančnosti obdelanih podatkov, ki jih dobimo iz senzorjev. Kot lahko bralec sklepa, je najbolj pomembna prva, ta pa je lahko z uporabo primerne tehnologijo zadovoljivo natančna (npr. laserski merilnik razdalje in kotov, natančen kompas). Torej, tudi če drugi dve popolnoma odpovesta, lahko uporabnik poišče NFC nalepko v svoji bližini (npr. v radiju 5 m) in pridobi informacijo o svoji lokaciji (ta je lahko opisna ali v koordinatah). Povedati moramo tudi, da je tretja odvisna tudi od druge, saj je izračun smeri vhodni podatek pri izračunu hitrosti in razdalje (Slika 16). Kot smo videli v tretjem poglavju, je za izračun smeri pomembna predvsem meritev jakosti magnetnega polja (poleg podatkov iz pospeškomera in žiroskopa, če zaradi natančnosti uporabimo senzorsko zlivanje), ki pa je lahko zaradi morebitnega močnega vpliva drugega magnetnega polja popolnoma napačna, zato je te anomalije potrebno sproti odkrivati in obveščati uporabnika. Ker se magnetni severni pol (proti kateremu kaže magnetna igla) vedno

spreminja, bi morali smeri, ki jih vpisujemo v zemljevid (v grafno podatkovno bazo), zaradi natančnosti vedno popravljati. Te spremembe so sedaj sicer majhne, lahko pa se zgodi, da s časom postanejo resen problem.

Naslednja zadeva, ki jo je potrebno izpostaviti, je razdelitev točk za določen zemljevid. V podatkovni strukturi Point imamo element Map, v katerega shranjujemo številko zemljevida, kamor spada posamezna točka (NFC nalepka). Iz tabele (Tabela 7) lahko vidimo, da je čas poizvedbe pri dolgih poteh (npr. če vsebujejo veliko točk) relativno dolg (nekaj sekund). Ne glede na to, da je čas poizvedbe v odvisnosti od števila vrnjenih točk na poti skoraj linearen, bi bilo smiselno omejiti največje število vrnjenih točk npr. na 300 za določen zemljevid in ponovno narediti poizvedbo za pot pri naslednjem zemljevidu. Kot vidimo, je prostora za izboljšave še kar nekaj, npr. interaktivni prikaz zemljevida, prikaz informacij v obliki obogatene resničnosti na zaslonu telefona, iskalnik po zanimivih točkah itd.

Priloge

Priloga A – Mobilna aplikacija

```
195    private class NdefReaderTask extends AsyncTask<Tag, Void, String> {
196        @Override
197        protected String doInBackground(Tag... params) {
198            Tag tag = params[0];
199
200            Ndef ndef = Ndef.get(tag);
201            if (ndef == null) {
202                return null;
203            }
204
205            NdefMessage ndefMessage = ndef.getCachedNdefMessage();
206            NdefRecord[] records = ndefMessage.getRecords();
207            for (NdefRecord ndefRecord : records) {
208                if (ndefRecord.getTnf() == NdefRecord.TNF_WELL_KNOWN && Arrays.equals(ndefRecord.getType(), NdefRecord.RTD_TEXT)) {
209                    try {
210                        return readText(ndefRecord);
211                    } catch (UnsupportedEncodingException e) {
212                        Log.e(TAG, "Unsupported Encoding", e);
213                    }
214                }
215            }
216            return null;
217        }
218
219        private String readText(NdefRecord record) throws UnsupportedEncodingException {
220            byte[] payload = record.getPayload();
221            // Get the Text Encoding
222            String textEncoding = ((payload[0] & 128) == 0) ? "UTF-8" : "UTF-16";
223            // Get the Language Code
224            int languageCodeLength = payload[0] & 0063;
225            return new String(payload, languageCodeLength + 1, payload.length - languageCodeLength - 1, textEncoding);
226        }
227
228        @Override
229        protected void onPostExecute(String responseFromNfc) {
230            if (responseFromNfc != null) {
231                try {
232                    if (onNfcReadingListener != null) {
233                        onNfcReadingListener.onNfcReading(responseFromNfc);
234                    }
235                } catch (Exception ex) {
236                    tvReceived.setText("Exception");
237                }
238            } else {
239                tvReceived.setText("No NFC result.");
240            }
241        }
242    }
243
244    public class HttpAsyncTask extends AsyncTask<String, Void, String> {
245        @Override
246        protected String doInBackground(String... urls) {
247            return GET(urls[0]);
248        }
249
250        @Override
251        protected void onPostExecute(String responseFromServer) {
252            try {
253                if (onServerResponseListener != null) {
254                    onServerResponseListener.onServerResponse(responseFromServer);
255                }
256            } catch (Exception e) {
257                e.printStackTrace();
258            }
259        }
260    }
261 }
```

Koda 6: Izsek kode MainActivity.java

```
328
329     @Override
330     public void onNfcReading(String result) {
331         HttpAsyncTask aTask = new HttpAsyncTask();
332         aTask.execute("http://192.168.111.91:1002/Service1.svc/getPoint?idPoint=" + result);
333         sendServerCommand = "POINT";
334     }
335
336     @Override
337     public void onServerResponse(String result) {
338         try {
339             if (sendServerCommand == "POINTS") {
340                 Type listType = new TypeToken<ArrayList<Point>>() {
341                 }.getType();
342                 points = new Gson().fromJson(result, listType);
343                 doLoadingPoints = false;
344                 // Remove startPoint from points
345                 points.remove(currentScannedPoint);
346                 addItemsOnSpinner(spEndPoint, points);
347                 addListenerOnSpinnerItemSelection();
348                 spEndPoint.setEnabled(true);
349                 tvDestination.setTextColor(Color.RED);
350             } else if (sendServerCommand == "POINT") {
351                 currentScannedPoint = new Gson().fromJson(result, Point.class);
352                 tvStartPoint.setText("1. You are here: " + currentScannedPoint.toString());
353                 tvStartPoint.setTextColor(Color.BLACK);
354                 isStarted = true;
355                 currentRelation = findCurrRelationByPoint(currentScannedPoint);
356                 prevRelation = findPrevRelationByPoint(currentScannedPoint);
357                 mDisplacement = new Displacement();
358                 // Getting points only with first scan
359                 if (doLoadingPoints) {
360                     HttpAsyncTask aTask = new HttpAsyncTask();
361                     aTask.execute("http://192.168.111.91:1002/Service1.svc/getAllPoints");
362                     sendServerCommand = "POINTS";
363                     btnStart.setEnabled(true);
364                     btnEnd.setEnabled(false);
365                 } else {
366                     // writeMeasurement(mReal, mMeasured);
367                 }
368             } else if (sendServerCommand == "PATH") {
369                 sPath = new Gson().fromJson(result, ShortestPath.class);
370                 isStarted = true;
371                 currentRelation = sPath.getRelations().get(0);
372             } else if (sendServerCommand == "MEASURE") {
373                 if (result.contains("OK"))
374                     Toast.makeText(context, "Measurement sent!", Toast.LENGTH_SHORT).show();
375                 else
376                     Toast.makeText(context, "Error by measurement!", Toast.LENGTH_SHORT).show();
377             }
378         } catch (Exception ex) {
379             Log.e("onServerResponse", "Error occured.", ex);
380             Toast.makeText(context, "Error occured - onServerResponse.", Toast.LENGTH_SHORT).show();
381         }
```

Koda 7: SensorActivity.java

```
1  package com.diploma.util;
2
3  public interface DiplomaListener {
4
5      public interface OnOrientationCalculatedListener {
6          public void onOrientationCalculated ();
7      }
8
9      public interface OnNfcReadingListener {
10         public void onNfcReading (String result);
11     }
12
13     public interface OnServerResponseListener {
14         public void onServerResponse (String result);
15     }
16 }
```

Koda 8: DiplomaListener.java

Priloga B – Strežniška aplikacija

```vb
Imports Diploma.Util
Imports Diploma.Data.Structures

Public Class Service1
    Implements IService1

    Public Sub New()
    End Sub

    Public Function GetPoint(ByVal idPoint As String) As Diploma.Data.Structures.Point Implements IService1.GetPoint
        Dim com As New Communication
        Return com.GetPointByPointId(Convert.ToInt32(idPoint))
    End Function

    Public Function GetAllPoints() As List(Of Point) Implements IService1.GetAllPoints
        Dim com As New Communication
        Return com.GetAllPoints()
    End Function

    Public Function GetAllLeafPoints() As List(Of Point) Implements IService1.GetAllLeafPoints
        Dim com As New Communication
        Return com.GetAllLeafPoints()
    End Function

    Public Function GetShortestPath(ByVal idStartPoint As String, ByVal idEndPoint As String) As ShortestPath Implements IService1.GetShortestPath
        Dim com As New Communication
        Return com.GetShortestPath(Convert.ToInt32(idStartPoint), Convert.ToInt32(idEndPoint))
    End Function

    Public Function GetShortestPathByNodeId(ByVal idStartPoint As String, ByVal idEndPoint As String) As ShortestPath Implements IService1.GetShortestPathByNodeId
        Dim com As New Communication
        Return com.GetShortestPathByNodeId(Convert.ToInt32(idStartPoint), Convert.ToInt32(idEndPoint))
    End Function

    Public Function GetDataUsingDataContract(ByVal composite As CompositeType) As CompositeType Implements IService1.GetDataUsingDataContract
        If composite Is Nothing Then
            Throw New ArgumentNullException("composite")
        End If
        If composite.BoolValue Then
            composite.StringValue &= "Suffix"
        End If
        Return composite
    End Function

    Public Function WriteMeasurement(ByVal realDistance As String, ByVal mDistance As String, ByVal type As String) As String Implements IService1.WriteMeasurement
        My.Computer.FileSystem.WriteAllText("C:\Meritve\meritve.txt", realDistance & " " & mDistance & " " & type & vbNewLine, True)
        Return "OK"
    End Function

End Class

```

Koda 9: WCF Service1.svc.vb

```vb
Imports Diploma.Data.Structures

<ServiceContract()>
Public Interface IService1

    <OperationContract()>
    <WebInvoke(Method:="GET", ResponseFormat:=WebMessageFormat.Json, RequestFormat:=WebMessageFormat.Json, UriTemplate:="getPoint?idPoint={idPoint}")>
    Function GetPoint(ByVal idPoint As String) As Point

    <OperationContract()>
    <WebInvoke(Method:="GET", ResponseFormat:=WebMessageFormat.Json, RequestFormat:=WebMessageFormat.Json, UriTemplate:="getAllPoints")>
    Function GetAllPoints() As List(Of Point)

    <OperationContract()>
    <WebInvoke(Method:="GET", ResponseFormat:=WebMessageFormat.Json, RequestFormat:=WebMessageFormat.Json, UriTemplate:="GetAllLeafPoints")>
    Function GetAllLeafPoints() As List(Of Point)

    <OperationContract()>
    <WebInvoke(Method:="GET", ResponseFormat:=WebMessageFormat.Json, RequestFormat:=WebMessageFormat.Json, UriTemplate:="GetShortestPath?idStartPoint={idStartPoint}&idEndPoint={idEndPoint}")>
    Function GetShortestPath(ByVal idStartPoint As String, ByVal idEndPoint As String) As ShortestPath

    <OperationContract()>
    <WebInvoke(Method:="GET", ResponseFormat:=WebMessageFormat.Json, RequestFormat:=WebMessageFormat.Json, UriTemplate:="GetShortestPathByNodeId?idStartPoint={idStartPoint}&idEndPoint={idEndPoint}")>
    Function GetShortestPathByNodeId(ByVal idStartPoint As String, ByVal idEndPoint As String) As ShortestPath

    <OperationContract()>
    <WebInvoke(Method:="GET", ResponseFormat:=WebMessageFormat.Json, RequestFormat:=WebMessageFormat.Json, UriTemplate:="WriteMeasurement?realDistance={realDistance}&mDistance={mDistance}&type={type}")>
    Function WriteMeasurement(ByVal realDistance As String, ByVal mDistance As String, ByVal type As String) As String

    <OperationContract()>
    Function GetDataUsingDataContract(ByVal composite As CompositeType) As CompositeType

    ' TODO: Add your service operations here

End Interface
```

Koda 10: IService1.vb

```vb
Imports System.Net
Imports System.IO
Imports Neo4jClient
Imports Neo4jClient.Cypher
Imports Neo4jClient.ApiModels.Cypher
Imports System.Collections.Generic
Imports Diploma.Data.Structures
Imports System.Text
Imports System.Runtime.Serialization.Json
Imports System.Web.Script.Serialization
Imports System.Runtime.CompilerServices

Public Class Communication

    Private Const DB_DATA_LOCATION As String = "http://localhost:7474/db/data"

    Public Function GetPointByPointId(ByVal id As Integer) As Point
        Dim client = New GraphClient(New Uri(DB_DATA_LOCATION))
        client.Connect()

        Dim query = client.Cypher.Start(New With {Key .p = All.Nodes}) _
                    .Where(Function(p As Point) p.Id = id) _
                    .Return(Of Point)("p")

        Return query.Results.ToList(0)
    End Function

    Public Function GetNodeIdByPointId(ByVal id As Integer, client As GraphClient) As Integer
        Dim query = client.Cypher.Start(New With {Key .p = All.Nodes}) _
                    .Where(Function(p As Point) p.Id = id) _
                    .Return(Of Node(Of Point))("p")
        Return query.Results.ToList(0).Reference.Id
    End Function

    Public Function GetPointJSONById(ByVal id As Integer) As String
        Dim client = New GraphClient(New Uri(DB_DATA_LOCATION))
        client.Connect()

        Dim query = client.Cypher.Start(New With {Key .p = All.Nodes}) _
                    .Where(Function(p As Point) p.Id = id) _
                    .Return(Of Point)("p")

        Return DeserializerPoint(query.Results.ToList(0))
    End Function

    Public Function GetPointByNodeId(ByVal idNode As Integer, client As GraphClient) As Point
        Dim query = client.Cypher.Start(New With {Key .p = "node(" & idNode & ")"}) _
                                 .Return(Of Point)("p")
        Return query.Results.ToList(0)
    End Function

    Public Function GetRelById(ByVal idRel As String, client As GraphClient) As RelationshipInstance
        Dim query = client.Cypher.Start(New With {Key .r = "rel(" & idRel & ")"}) _
                                 .Return(Of RelationshipInstance(Of ConnectedRelationship))("r")
        Return query.Results.ToList(0)
    End Function
```

Koda 11: Izsek kode Communication.vb

Priloga C – Aplikacija za vnos zemljevida

```
7   Imports System.Text
8
9   Public Class Form1
10
11      Private i = 0
12      Dim _PointList As New List(Of Point)
13
14      ' POST a JSON string
15      Public Sub POST(uri As Uri, jsonContent As String)
16          Dim encoding As New System.Text.UTF8Encoding()
17          Dim request As HttpWebRequest = DirectCast(WebRequest.Create(uri), HttpWebRequest)
18          request.Method = "POST"
19          Dim byteArray As [Byte]() = encoding.GetBytes(jsonContent)
20          request.ContentLength = byteArray.Length
21          request.ContentType = "application/json"
22
23          Using dataStream As Stream = request.GetRequestStream()
24              dataStream.Write(byteArray, 0, byteArray.Length)
25              dataStream.Close()
26          End Using
27          Dim length As Long = 0
28          Try
29              Using response As HttpWebResponse = DirectCast(request.GetResponse(), HttpWebResponse)
30                  length = response.ContentLength
31                  Dim sr = New StreamReader(response.GetResponseStream())
32                  Label1.Text = sr.ReadToEnd()
33              End Using
34          Catch ex As WebException
35              Label1.Text = ex.Message
36          End Try
37      End Sub
38
39      Private Sub Create() ...
50
51      Private Sub GraphFromGrid(client As GraphClient) ...
56
57      Private Sub OneWayGraph(client As GraphClient) ...
167
168     Private Sub GannaGraph(client As GraphClient) ...
242
243     Private Sub CreateBigGraph(client As GraphClient) ...
337
338     Private Sub GetShortestPath() ...
350
351     Private Sub BtnCreateGraph_Click(sender As Object, e As EventArgs) Handles BtnCreateGraph.Click ...
354
355     Private Sub BtnResetPoints_Click(sender As Object, e As EventArgs) Handles BtnResetPoints.Click ...
359
360     Private Sub BtnResetRelations_Click(sender As Object, e As EventArgs) Handles BtnResetRelations.Click ...
364
365     Private Sub BtnResetGraph_Click(sender As Object, e As EventArgs) Handles BtnResetGraph.Click ...
377
378     Private Sub BtnCreatePoints_Click(sender As Object, e As EventArgs) Handles BtnCreatePoints.Click ...
400
401     Private Sub BtnCreateRelations_Click(sender As Object, e As EventArgs) Handles BtnCreateRelations.Click ...
404
405     Private Sub DataGridViewRelations_RowsAdded(sender As Object, e As DataGridViewRowsAddedEventArgs) Handles DataGridViewRelations.RowsAdded ...
417 End Class
418
```

Koda 12: Izsek kode aplikacije za vnos zemljevida

Seznam slik

Slika 1: Zgodovina razvoja NFC tehnologije [4] 7
Slika 2: Primer iniciatorja in tarče 8
Slika 3: Komunikacija med mobilnim telefonom in nalepko 9
Slika 4: Komunikacija med dvema mobilnima telefonoma 9
Slika 5: Komunikacija med mobilnim telefonom in bralnikom 9
Slika 6: Format za izmenjavo podatkov, ki jo uporablja NFC tehnologija 10
Slika 7: Blok diagram Mifare Ultralight C nalepke 10
Slika 8: Organizacija pomnilnika Mifare Ultralight C nalepke 10
Slika 9: Primer zemljevida z vrisanimi potmi 12
Slika 10: Določanje kota oz. smeri na povezavi 12
Slika 11: Telefon sporoči trenutno lokacijo uporabnika 13
Slika 12: Navigacija 14
Slika 13: Sporočilo ob prispetju na cilj 14
Slika 14: Graf z usmerjenimi povezavami 15
Slika 15: Z rdečo barvo označena najkrajša pot grafa G 16
Slika 16: Senzorsko zlivanje 18
Slika 17: Dijkstrov algoritem v psevdokodi [6] 20
Slika 18: Arhitektura sistema 23
Slika 19: Zemljevid s 300 generiranimi točkami 25
Slika 20: TCP/IP sklad prikazuje potovanje poizvedbe preko omrežja. 25
Slika 21: Podatkovna struktura Point 26
Slika 22: Podatkovna struktura Relation 27
Slika 23: Podatkovna struktura ShortestPath 28
Slika 24: UML razredni diagram mobilne aplikacije 29
Slika 25: Graf odvisnosti med strežniškimi moduli 30
Slika 26: Aplikacija za vnos enostavnega zemljevida v grafno podatkovno bazo 33
Slika 27: Deklinacija – kot med geografskim (Ng) in magnetnim (Nm) severom 34

Seznam tabel

Tabela 1: Frekvenčna območja in razdalje delovanja 8
Tabela 2: Primerjava HF RFID in NFC 8
Tabela 3: Poizvedba posamezne točke in rezultat 23
Tabela 4: Poizvedba vseh točk v grafu in rezultat 24
Tabela 5: Poizvedba po vseh končnih točkah (listih) in rezultat 24
Tabela 6: Poizvedba po najkrajši poti in rezultat 24
Tabela 7: Rezultati meritev 25
Tabela 8: Opis podatkovne strukture Point 27
Tabela 9: Opis podatkovne strukture Relation 27
Tabela 10: Opis podatkovne strukture ShortestPath 28
Tabela 11: Čas izvajanja poizvedbe pri relacijski in grafni podatkovni bazi 31
Tabela 12: Meritve magnetnega polja 36

Seznam grafov

Graf 1: Linearni pospešek, hitrost in razdalja v odvisnosti od časa 17
Graf 2: Bistveno izboljšan rezultat z uporabo senzorskega zlivanja 19
Graf 3: Primerjava med meritvami 36

Seznam izsekov izvorne kode

Koda 1: Izračun rotacijske matrike, iz katere izračunamo orientacijo telefona. 22
Koda 2: Poizvedba posamezne točke po Id s Cypher poizvedbo 32
Koda 3: Poizvedba po vseh točkah s Cypher poizvedbo 32
Koda 4: Poizvedba po vseh končnih točkah (listih) 32
Koda 5: Poizvedba po najkrajši poti s POST (angl. HTTP Request) 32
Koda 6: Izsek kode MainActivity.java 43
Koda 7: SensorActivity.java 44
Koda 8: DiplomaListener.java 44
Koda 9: WCF Service1.svc.vb 45
Koda 10: IService1.vb 45
Koda 11: Izsek kode Communication.vb 46
Koda 12: Izsek kode aplikacije za vnos zemljevida 47

Viri

[1] (2013) Satellite navigation. Dostopno na: http://en.wikipedia.org/wiki/Satellite_navigation

[2] S. G. e. al., "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks, IEEE in Signal Processing Magazine, vol. 22, no. 4, pp. 70–84, 2005.

[3] (2014) Near Field Communication. Dostopno na:
http://sl.wikipedia.org/wiki/Near_Field_Communication

[4] K. O. B. O. Vedat Coskun, Near Field Communication (NFC): From Theory to Practice, John Wiley & Sons Canada, Limited, 2012.

[5] (2009) A. Abdelkader. Dostopno na:
http://the-lost-beauty.blogspot.com/2009/12/simulation-and-kalman-filter-for-3rd.html

[6] (2014) Dijkstra's algorithm. Dostopno na: http://en.wikipedia.org/wiki/Dijkstra's_algorithm#Pseudocode

[7] (2013) Motion Sensors - Android Developer's Guide. Dostopno na: http://developer.android.com/guide/topics/sensors/sensors_motion.html

[8] J. W. &. E. E. Ian Robinson, Graph Databases, O'Reilly, 2013, str. 8–10.

[9] (2014) Neo4j. Dostopno na:
http://docs.neo4j.org/chunked/milestone/index.html

[10] (2013) Earth's magnetic field. Dostopno na: http://en.wikipedia.org/wiki/Earth's_magnetic_field

[11] (1999) J. Zupan, Zemeljsko magnetno polje. Dostopno na: http://www.kvarkadabra.net/pojavi/teksti/magnet_zemlja.htm

[12] (2013) V. Kopytoff, Stores Sniff Out Smartphones to Follow. Dostopno na: http://www.technologyreview.com/news/520811/stores-sniff-out-smartphones-to-follow-shoppers/